RasPi

DESIGN
BUILD
CODE

36

Get hands-on with your Raspberry Pi

USE PI AND AMAZON ECHO TO

AUTOMATE YOUR HOME



+

BUILD A **DOOMSDAY** SWITCH

Plus Monitor your servers with Pi

Welcome



"Let there be light!" It's hard not to feel godlike when issuing such a command and while this issue of Raspi can't offer you omniscience or infinite wisdom, it will show you how to control your household lighting via your voice, divine or otherwise. Swipe along to our main feature and learn how to combine your Pi with an Amazon Echo to control the Pimoroni home lighting kit, Mote. We'll show you how to create an unpublished skill that will provide some miraculous utterances for changing the colour of your lighting and turning it on and off. Hallelujah.

Also in this issue you can build a doomsday switch to nuke your precious data in case of a breach and combine Python and Pi to keep an eye an your servers. Swipe on!

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi



Editor

From the makers of

LinuxUser & Developer

Join the conversation at...

@linuxusermag | Linux User & Developer | linuxuser@futurenet.com

Contents

Control lighting with the Pi and Amazon Echo
Control Pimoroni's home lighting kit for the Raspberry Pi

Pi Project: PSPi V2
Adam Seamster's PSPi 2 provides a great gaming experience

Make a Raspberry Pi doomsday switch
How to erase your home folder in case of emergency

Turn your Pi into a Tor proxy with Onion Pi
Create a wireless access point for the Tor network

Encrypt your Raspberry Pi home folder
Bring military-grade encryption to your Pi

Monitor your servers with a Raspberry Pi
Avoid crashes and data loss with handy monitoring tools

Talking Pi
Your questions answered and your opinions shared

Control lighting with the Pi and Amazon Echo

Control Pimoroni's home lighting kit for the Raspberry Pi through the Alexa Skills kit

In 2016, Amazon brought the Echo (£150) and Echo Dot (£50) devices to the UK market, providing a voice assistant for your home. The Alexa service provides lots of built-in features such as radio and music streaming, creating and editing your shopping lists, weather updates and many other custom skills provided by third parties. Skill builders such as Uber and Capital One publish their skills for the general public, which means they go through a vetting process similar to that of the Apple App Store. For this tutorial, we will be creating an unpublished skill for our own Echo or Echo Dot to control the Pimoroni home lighting kit called Mote.

Our skill will provide utterances for changing the colour of our LEDs like "Alexa, ask Mote change to blue" and will let us turn them off by saying "Alexa, tell mote turn off."

THE PROJECT ESSENTIALS

Github repository: https://github.com/alexellis/motephat-alexa

Pimoroni mote-phat and accessories (pimoroni.com)

Soldering iron, flux and solder

01 Prepare your hardware

Prepare your mote-phat add-on board by attaching and soldering its 40-pin female header, included in the packaging. If you're using a Pi Zero you will also need to solder a 40-pin male header before continuing.



02 Set up the base system

Flash a new SD card with Raspbian Jessie Lite, making sure to create a file in the boot partition called 'ssh'; this will let us connect over SSH remotely and copy/paste commands without needing UI packages or a screen. Once plugged in, your Pi will be accessible via ssh pi@raspberrypi.local.

03 Responding to Alexa – Lambda or HTTPS

Alexa can either invoke code over an HTTPS endpoint (web-service) or via a Lambda function, which is a piece of code uploaded to Amazon's AWS service and invoked on demand. For our tutorial, we'll set up our own HTTPS

endpoint from our Pi to the public internet with the ngrok tool. Download and unzip ngrok for Linux ARM into /usr/bin` from https://ngrok.com/download

04 Install Docker

Docker is a packaging and runtime system that allows us to build, ship and run software easily. Run these two commands then reconnect over SSH.

```
  ```
 # curl -sSL get.docker.com | sh
 # sudo usermod pi -aG docker
  ```
```

Now clone the Github repository and build the Docker image (this will take some time):

```
  ```
```

**Below** The resulting Docker image contains everything needed for our application in an isolated package.

Branch: master ▾ **motephat-alexa** / **Dockerfile** Find file | Copy path

**alexellis** Merge removal of entrypoint b3214b2 13 days ago

1 contributor

21 lines (17 sloc) | 495 Bytes Raw | Blame | History

```
1 FROM resin/rpi-raspbian
2 MAINTAINER alexellis2@gmail.com
3
4 RUN apt-get update \
5 && apt-get install git python-dev python-pip gcc \
6 && git clone https://github.com/pimoroni/mote-phat \
7 && pip install flask \
8 && cd mote-phat/library && python setup.py install \
9 && apt-get -qy remove python-dev gcc \
10 && rm -rf /var/lib/apt/lists/*
11
12 WORKDIR /root/
13 RUN mkdir /root/alexa/
14 WORKDIR /root/alexa/
15 COPY mote.py .
16 COPY app.py .
17 EXPOSE 5000
18 ENTRYPOINT []
19 CMD ["python", "./app.py"]
20
```
```

```
  # apt-get update && apt-get -qy install git
jq
    #  git  clone  https://github.com/alexellis/
motephat-alexa
  # docker build -t alexamote .
  ```
```

The resulting Docker image contains everything needed for our application in an isolated package.

## 05 Start the code with Docker

Our project's code is packaged with all its dependencies into a single container. We can now run that in the background and open the ngrok HTTPS tunnel to the internet. The flag -p tells Docker to expose the port for our web server code that talks to Alexa. The -d flag tells the service to run in the background.

```
 # docker run --name mote --privileged -d -p
5000:5000 alexamote
 # ngrok http 5000 > /dev/null &
 # curl localhost:4040/api/tunnels | jq -r
```

```
~ — pi@mote: ~/dev/motephat-alexa — ssh u2
pi@mote:~/dev/motephat-alexa $ docker run --name mote --privileged -d -p 5000:5000 alexamo
te
3c1325c7b0d39b0d537c37801ff74618ffa835644abd1b0f48df081e5bebc82c
pi@mote:~/dev/motephat-alexa $ ngrok http 5000 > /dev/null &
[1] 2428
pi@mote:~/dev/motephat-alexa $ curl localhost:4040/api/tunnels | jq -r ".tunnels[1].public
_url "
 % Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
100 755 100 755 0 0 17730 0 --:--:-- --:--:-- --:--:-- 18414
pi@mote:~/dev/motephat-alexa $ curl -X POST -H "Content-type: application/json" -d @colour
sample.json https://9438e1a8.ngrok.io
{"version": "1.0", "response": {"outputSpeech": {"text": "OK, changing to red", "type": "P
lainText"}, "shouldEndSession": true, "card": {"content": "OK, changing to red", "type": "
Simple", "title": "Colour change"}}, "sessionAttributes": {}}pi@mote:~/dev/motephat-alexa
$
pi@mote:~/dev/motephat-alexa $
pi@mote:~/dev/motephat-alexa $
```

## Natural Language Parsing (NLP)

The Alexa service sends recordings from your device to the Amazon cloud where NLP (Natural Language Parsing) breaks the words down into intents and slots. A sample phrase such as "Alexa, what is the weather for London tomorrow?" would be parsed as an intent of "FindWeather" with two slots of: Date=tomorrow and Location=London, UK. Sample utterances help cover the many different ways we can say the same thing in a language.
```

```
".tunnels[1].public_url "
  ```
```

Take note of your 'public_url' beginning with https. This changes every time the ngrok process starts.
If you want to stop the mote-phat process later on you can type in docker rm -f mote or docker ps to view its status.

## 06 Test the endpoint

Once you have your HTTPS URL from ngrok, then you can test everything out by sending in a request just like the one the Alexa SDK creates. We have captured two samples and saved them in the Git repository.
Test going red:

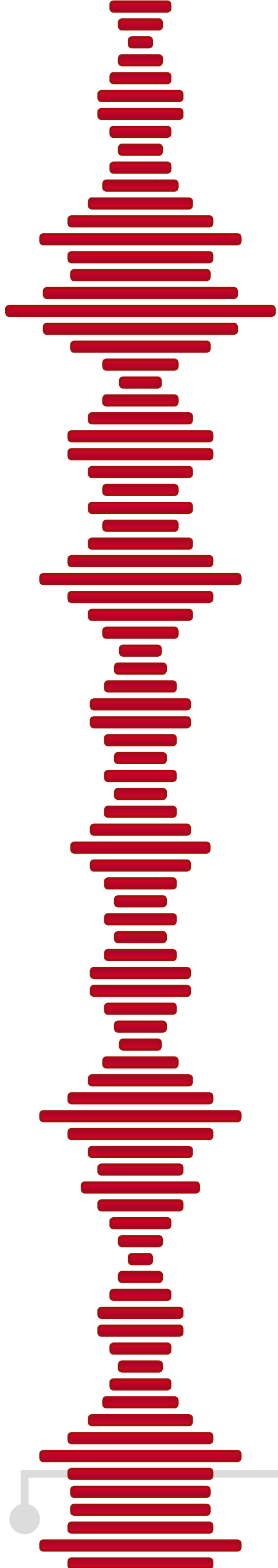
```
  ```
  # curl -X POST -H "Content-type: application/
json"  -d  @coloursample.json  https://c00738f6.
ngrok.io
  ```
```

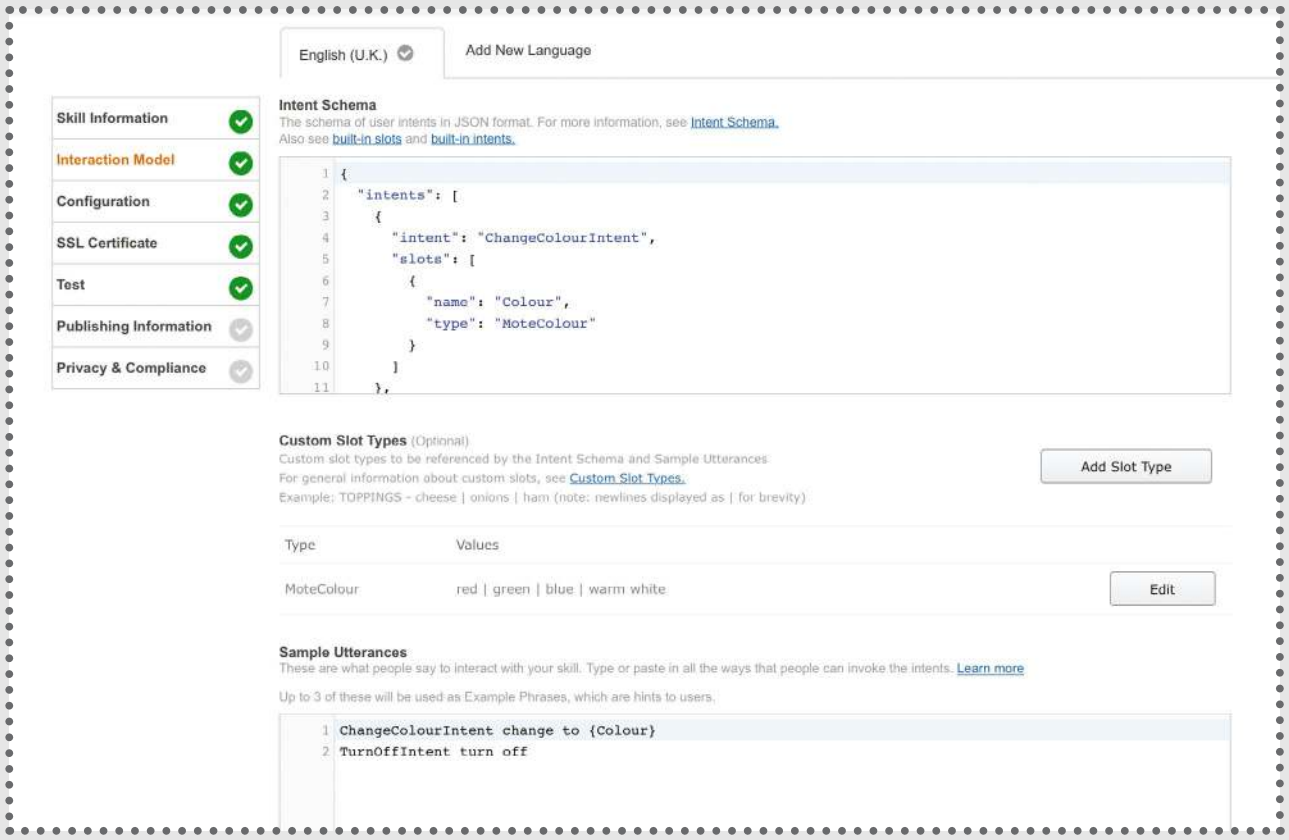
Test turning the lights off:

```
  ```
  # curl -X POST -H "Content-type: application/
json"  -d  @coloursample.json  https://c00738f6.
ngrok.io
  ```
```

## 07 Create an Alexa skill

Head over to https://developer.amazon.com/myapps.html and click Alexa>Alexa Skills Kit>Get Started. You may need to register for this step and provide billing information for any purchases you want to make.


```



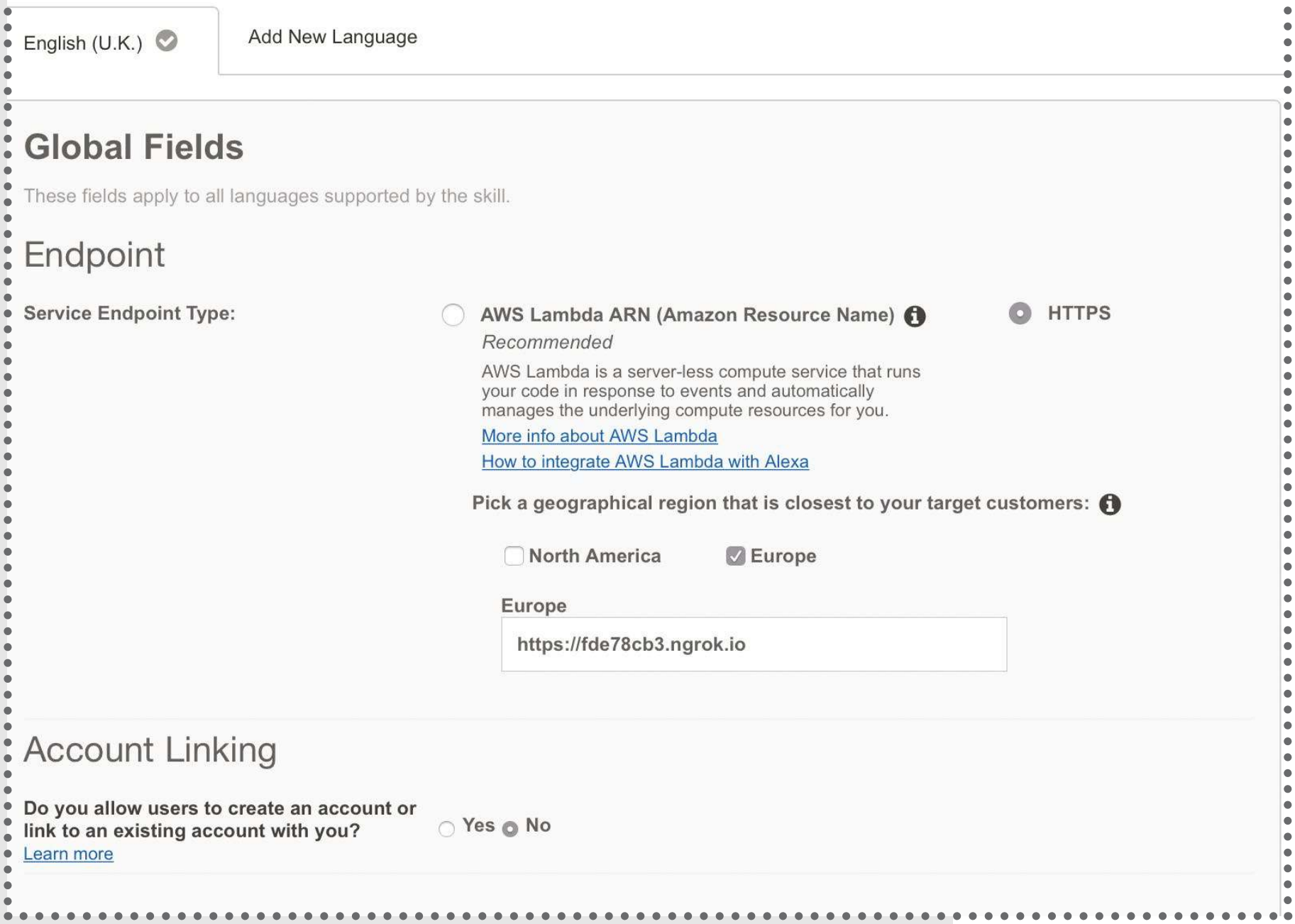
Left The custom slot helps Alexa by providing a list of all the things you could say

Click Add a New Skill>English UK and type in 'mote' for the name and invocation name fields. For the intent schema, copy/paste 'speechAsssets/intentSchema.json' and for sample utterances 'speechAsssets/sampleUtterances.txt'. You must also add a custom slot called colour with the values red/green/blue on separate lines. The custom slot helps Alexa by providing a list of all the things you could say to her – it's like a parameter in coding.



08 Point Alexa to your HTTPS endpoint

Under the configuration tab of your Alexa Skill, click service endpoint type: HTTPs. Then select your nearest



region (this will be Europe in the UK) and paste in the ngrok URL from earlier.

Now click 'My development endpoint has a certificate from a trusted certificate authority' on the SSL Certificate tab.

On the Test tab you can type in sample utterances such as "change to blue" or "turn off" – when you click 'Ask mote' a message will be transmitted to your Pi from Alexa's online service bypassing the Echo/Dot.

09 Talk to your Echo

If everything worked you will be able to talk to your Echo/Dot. Simply say "Alexa, ask mote change to red" or "Alexa, ask mote to turn off". For dimming the brightness level you can take inspiration

from Alex's Christmas Tree hack's source-code at: http://blog.alexellis.io/christmas-iot-tree/

```
30          if post_data["request"]["intent"]["name"] == "TurnOffIntent":
31                  response = get_response("Turning off", "OK")
32          else:
33                  response = get_response("OK setting desired colour", "OK")
34                  slot_colour = post_data["request"]["intent"]["slots"]["Colour"]["value"]
35                  if not slot_colour in ["red", "green", "blue"]:
36                          response = get_response("Can only set red, green or blue", "Error")
37                  else:
38                          red = 0
39                          green = 0
40                          blue = 0
41
42                          if slot_colour == "red":
43                                  red = 255
```

10 Next steps

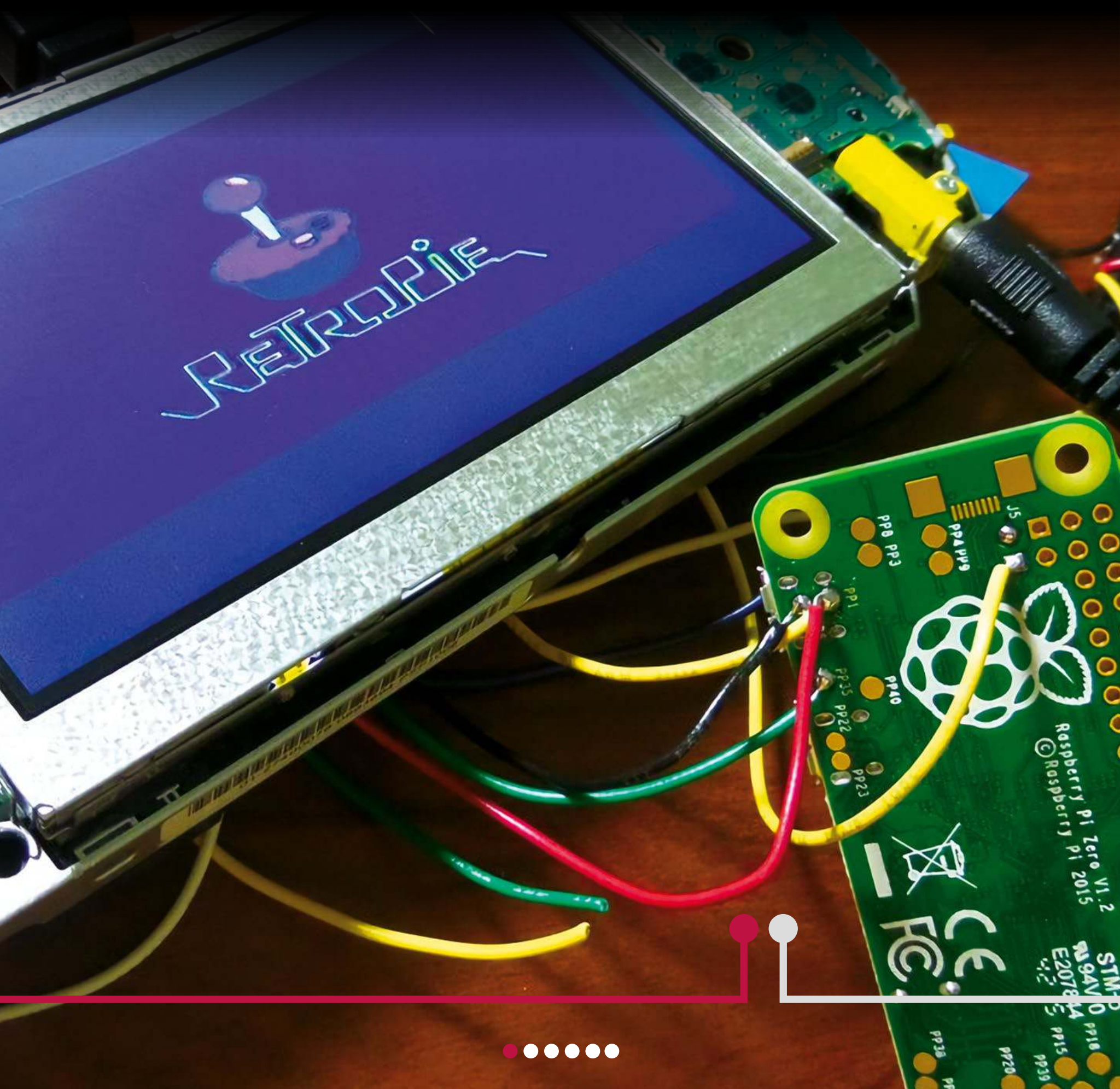
Now that you have created your first skill, maybe you can think of some ways to extend it or to apply it to other hardware projects? We think dimming the light could be useful and it should be easy to add other colours. If you want to know more about Docker, check out the boxout, the Dockerfile on the articles GitHub and Alex's beginner tutorials at: http://blog.alexellis.io/tag/raspberry-pi/ ■

Docker

Docker is a game-changer for packaging, deploying and running software. Each time you build software, Docker creates an image with its own root filesystem and network addresses. Inside a Docker container it feels exactly like a full virtual machine, except faster, because a container is regular process with some advanced syscalls applied for security and isolation.
The Docker CLI is intuitive for Linux users with commands like 'docker ps', 'docker run', and 'docker kill'. The 'docker build' command uses a Dockerfile, which is similar to a Makefile.

PSPi V2

Want a cost-effective way to improve your gaming experience? Adam Seamster's PSPi 2 project will be perfect for you



Where did the idea of the PSPi project originally come from?

I grew up playing Mario and Zelda, and then finally Pokémon in the late Nineties. Once a year or so I dust off the SNES or Gameboy Color and play one of the games through. This year though has been a busy one for retro gaming. Every few weeks someone is gutting an original Gameboy or DS and modifying it for the Pi Zero or Pi 3. Some are even designing their own cases. These projects have been intriguing, and they inspired me to build one that fits my needs a little better. My focus in all my projects is in keeping the cost low, since I prefer to spend time on a project rather than money. This meant finding an existing case for the hardware that was inexpensive, had a larger LCD to suit my preferences, and that had the right buttons to play console and handheld games. The ability to connect external USB devices was also important. After all, the Pi is a mini computer, and the ability to use it as one makes it that much more useful. A broken PSP was the perfect fit.

The Pi Zero plays a big role in the development of the PSPi, could you tell us how easy you found it to work with? Was the choice of the Pi Zero purely down to its size?

The Pi Zero was very easy to work with once a few basics were understood. This is not a device that gets plugged in and is ready to go. It requires time and effort. For this project specifically it was size that made the decision on the Pi version used. For projects in general though, it's all about cost, and this project



Adam Seamster is a long time Raspberry Pi tinkerer, with a history of breathing new life into aging handheld consoles. All of his work is showcased over on www.othermod.com.

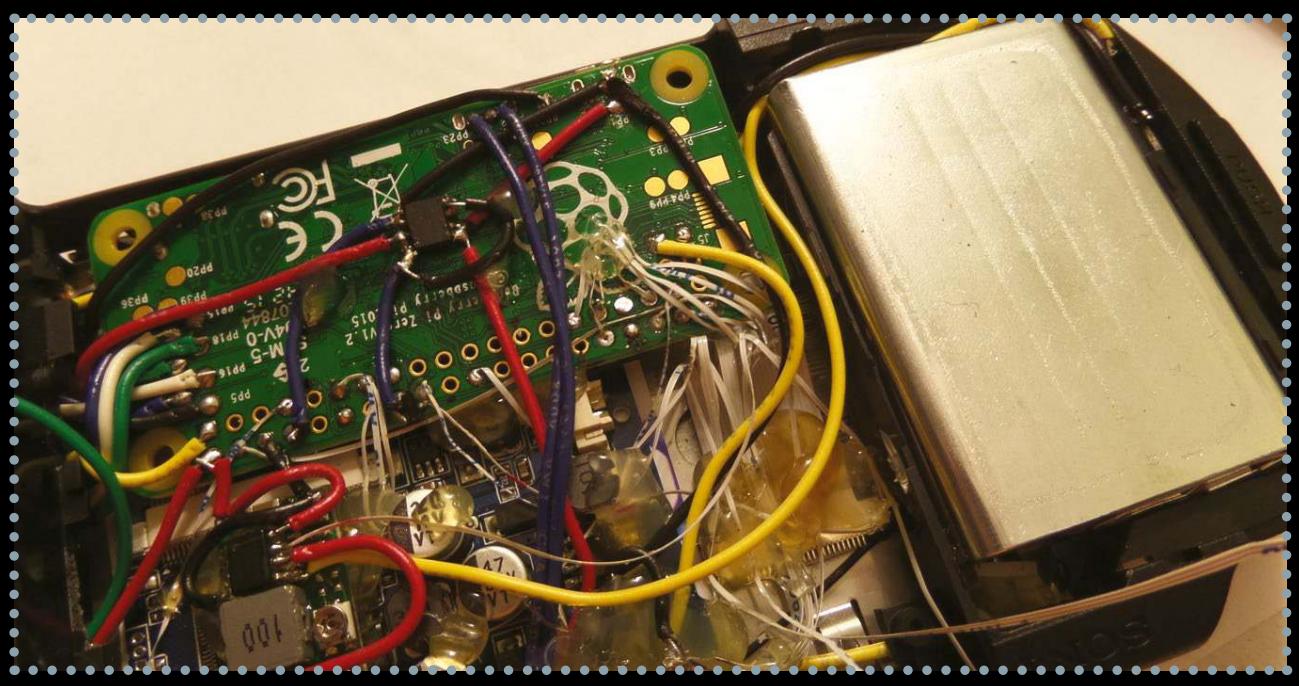
probably wouldn't have happened if the Pi Zero's cost was $30. I've been a tinkerer since I was a child, and having this $5 device with endless possibilities is very exciting. This will not be my last project using the Pi Zero.

Are you using a regular computer monitor for the display?

Yep, and the bezels are actually just the bezels from the computer monitor as well. It just happened to look right so I kept that. It's just a monitor that I bought at a thrift store called Value Village for about four, five dollars, something like that. It's not the newest thing ever, but it works more than well enough for what I'm doing.

What are they key differences between the original PSPi and the PSPi 2 that you launched recently?

The original PSPi was an experiment and was done without any community involvement. I toiled away for a few hours after work every day for about two months before finally arriving at a playable handheld. Everything was very hacked together because there wasn't a process yet. The design was being made



Left Adam created a soft power on and off circuit in the PSPi 2, to make it easier for gamers to turn the small device on or off quickly

THE PROJECT ESSENTIALS

Raspberry Pi Zero

4.3" composite LCD screen

IRF7319 dual MOSFET

LM393 voltage comparator

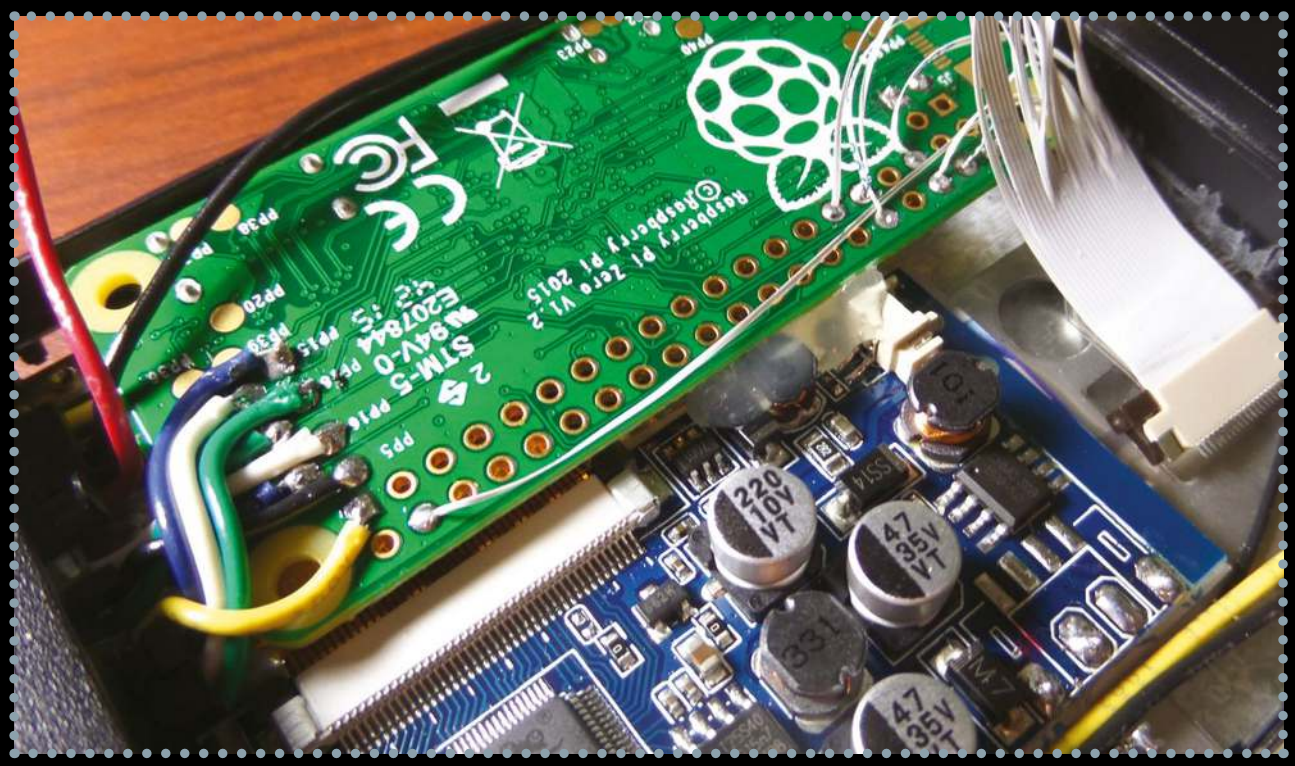
Lithium charge controller and power board

Audio amplifier

Lithium batteries

FPC connectors and cables

BAT54C dual diode



Left Each individual control on the PSP had to be soldered into place at the exact location needed to register a movement or action

up as I went along and it shows that there was re-engineering done a few times. Although everyone loved the 'rat's nest' of the original, the second is much cleaner and organised. The second PSPi is almost identical in functionality, but it is more clean and organised. Lessons learned on the first one were used to make changes to the second, including the use of a different LCD and better quality audio components. One major addition to PSPi 2 is going to be the analog joystick, which was present in Version 1 but not actually functional.

Did you experience any major development problems throughout the project? Any specific areas that caused you a few headaches?

Every part of this project has been a learning experience. I knew how the finished device needed to look and function, but had little direction on how to get there. Luckily there was plenty of time for research and planning when the project started because the Pi Zero was in short supply. This gave time to research



different hardware aspects of the audio filter, GPIO controls, and the composite LCD interface. This didn't prevent issues though. It took trial and error to find an audio circuit with reasonable sound quality and the power circuit had to be designed specifically for the Pi. Also, the hardware is only half of any build like this, and the software was just as complicated. Sharing this information is one reason I created the otherMod.com website. I've created tutorials for each part of the project as progress is made, hopefully allowing it to be recreated by those with less experience or less time to do the extensive research.

How easy would it be for our readers to recreate your PSPi 2 project? Do you have any tips for those willing to take it on?

This project is difficult to recreate. It contains many separate circuits and components that must be individually connected to each other, so an understanding of basic electronics is helpful and some

Like it?

If you find Adam's project a little complex, then you'll be able to find some more Pi-based gaming builds all over the place. If you want to create your own, you can either follow Adam's tutorials on his site, or search out the RetroPie website.

soldering experience is required. This is not including the work on the software side, which was even more difficult than the hardware at times. My goal with the PSPi was to make it as cheap as possible, leading to increased complexity and difficulty because of how custom every part of it was. The site's tutorials are meant to help with much of this, but this is still a project that many will find daunting. That being said, I'm working on changing it. I'm working with the community in designing a custom circuit board to interface with the Pi Zero and the PSP. The goal is to combine most of the custom circuits onto one board, and this board will require minimal soldering to the Pi Zero. Hopefully this will make the project available to many others that want a PSPi of their own.

What's the future of the PSPi project? Do you think there's a third version somewhere on the horizon?

This is only the beginning, and Version 3 will be hitting the site in the near future. Version 2 was the first in-progress project to be exposed to the community, and others in the community have responded with great ideas for changes and additions that will make their way into the next version. I never expected the project to become as popular as it did, and I'll keep it going until we reach a point where anyone can make one for themselves.



Further reading

Adam has been kind enough to detail the whole build process of his project on his site (othermod.com). You'll also be able to find a plethora of tutorials that can help you get the basics of recreating your own PSPi project.

Make a Raspberry Pi doomsday switch

Keep your data safe with a handy ‘nuke’ password to erase your home folder in case of emergency

THE PROJECT ESSENTIALS

Suitable for all models of Raspberry Pi

If you’re worried about the somewhat Orwellian notion of forced disclosure of passwords, this project posits a rather radical solution to the dilemma by creating a second password for your user account, which, instead of logging you in, will nuke your home folder using special tools.

While this is simple to set up, do make sure to back up your personal data to a safe place before going ahead. Also bear in mind that anyone with physical access to your machine may seize it before you have a chance to flip this kill switch.

01 Create your new user account

Although you most likely will already have a user account on the Pi, create a new one for this project by opening Terminal on your Pi or connecting via SSH and running the command

```
sudo adduser name
```

```
pi@raspberrypi ~> sudo adduser alice
Adding user `alice' ...
Adding new group `alice' (1001) ...
Adding new user `alice' (1001) with group `alice' ...
Creating home directory `/home/alice' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
        Full Name []: Alice Smith
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
pi@raspberrypi ~> sudo adduser alice sudo
Adding user `alice' to group `sudo' ...
Adding user alice to group sudo
Done.
pi@raspberrypi ~>
```

Add your new user as an Administrator with

```
sudo adduser name sudo
```

Substitute 'name' with your chosen username.

Above Choose your normal login password. Fill in the other fields or just press return to leave blank

02 Create your Nuke script

You should stay logged into the 'Pi' user for now and run the following command:

```
sudo nano /etc/security/security.sh
```

In the new window, paste the following:

```
#!/bin/bash
read password

# If the username and password match...
if [ “$PAM_USER” = “name” ] && [
“$password” = “nukepassword” ]
```

```
  then
  #Begin Nuke Process
  echo “Nuke is starting.”
  #Securely erase the home folder
  srm -rvvv /home/name/
  echo “Home folder has been erased.”
  #Overwrite the /home folder with random
data
  #sfill /home
  echo “Home folder has been overwritten”
  #Clean RAM memory
  #smem
  echo “RAM is clean”
  echo “User data has been nuked.”
  fi

  exit 0
```

03 Modify the script

In Line 5, substitute ‘name’ and ‘nukepassword’ for the username of your new account and the desired nuke password. Make sure this is different to your current one. **Change ‘srm -rvvv /home/name/’** to the path of your real home folder.



Below Press Ctrl + X, then Y, then return to save and exit

04 Run nuke script on login

Make your nuke script executable with the

command…

```
sudo chmod a+x security.sh
```

Next run…

```
sudo nano /etc/pam.d/common-auth
```

…to open the Pluggable Authentication Modules (PAM). Find the line starting **'auth [success=1…'** and change this to 'auth **[success=2…'.** Immediately below this line, paste the following:

```
auth optional pam_exec.so expose_authtok log=/tmp/pam.log /etc/security/security.sh
```

05 Install secure delete tools

Run the command…

Below Type 'man srm' for more information on how securely your data is erased.

```
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
auth    [success=2 default=ignore]      pam_unix.so nullok_secure
auth    optional        pam_exec.so debug expose_authtok log=/tmp/pam.log /etc/
                        [ Read 27 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

```
sudo apt-get install secure-delete
```

…to install the tools necessary to erase your home folder securely.

Substitute 'name' with your chosen username.

06 Migrate your data (Optional)

If you previously had personal data in another user

```
pi@raspberrypi ~> sudo apt-get install secure-delete
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  secure-delete
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 67.9 kB of archives.
After this operation, 112 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ jessie/main secure-delete ar
hf 3.1-6 [67.9 kB]
Fetched 67.9 kB in 0s (70.9 kB/s)
Selecting previously unselected package secure-delete.
(Reading database ... 128013 files and directories currently installed.)
Preparing to unpack .../secure-delete_3.1-6_armhf.deb ...
Unpacking secure-delete (3.1-6) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up secure-delete (3.1-6) ...
pi@raspberrypi ~>
```

account, you should take this chance to move data across from that account to another from your backup drive. If you wish to delete the originals, do so using the new secure-delete tools, for instance:

```
'srm -r /home/bob/Pictures'
```

07 Test your new account

Reboot your Pi and log into your new user account using the normal login password. Check that your files are where you need them.

08 Test your nuke switch

If your data is backed up, there's no harm checking

Nuclear deterrent

A nuke switch may appeal to your sense of melodrama but it's likely that if an unscrupulous person seizes your Pi, they will make a copy of the contents of the SD card before trying to enter the password. Ideally, you should encrypt your home folder before using this script. Even if it does work, the script cannot delete itself so it will be obvious to everyone that you have 'gone nuclear'.

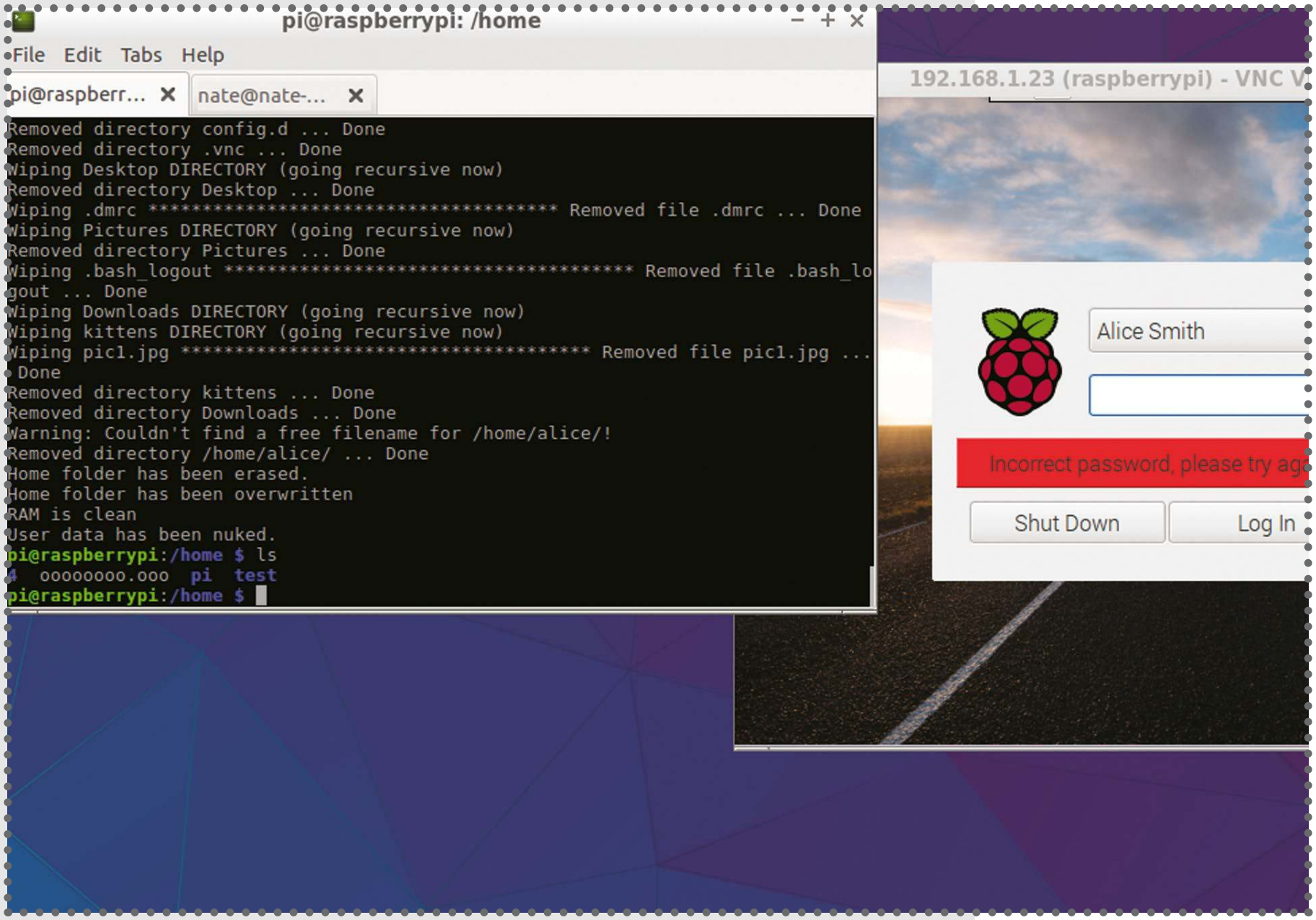
your nuke password works. Reboot the Pi once again. Select your new username and enter the nuke password. The system will hang while it removes your files.

09 Check nuke logs

You can still connect to the Pi via SSH while the nuke script is running. Use the command…

```
cat /tmp/pam.log
```

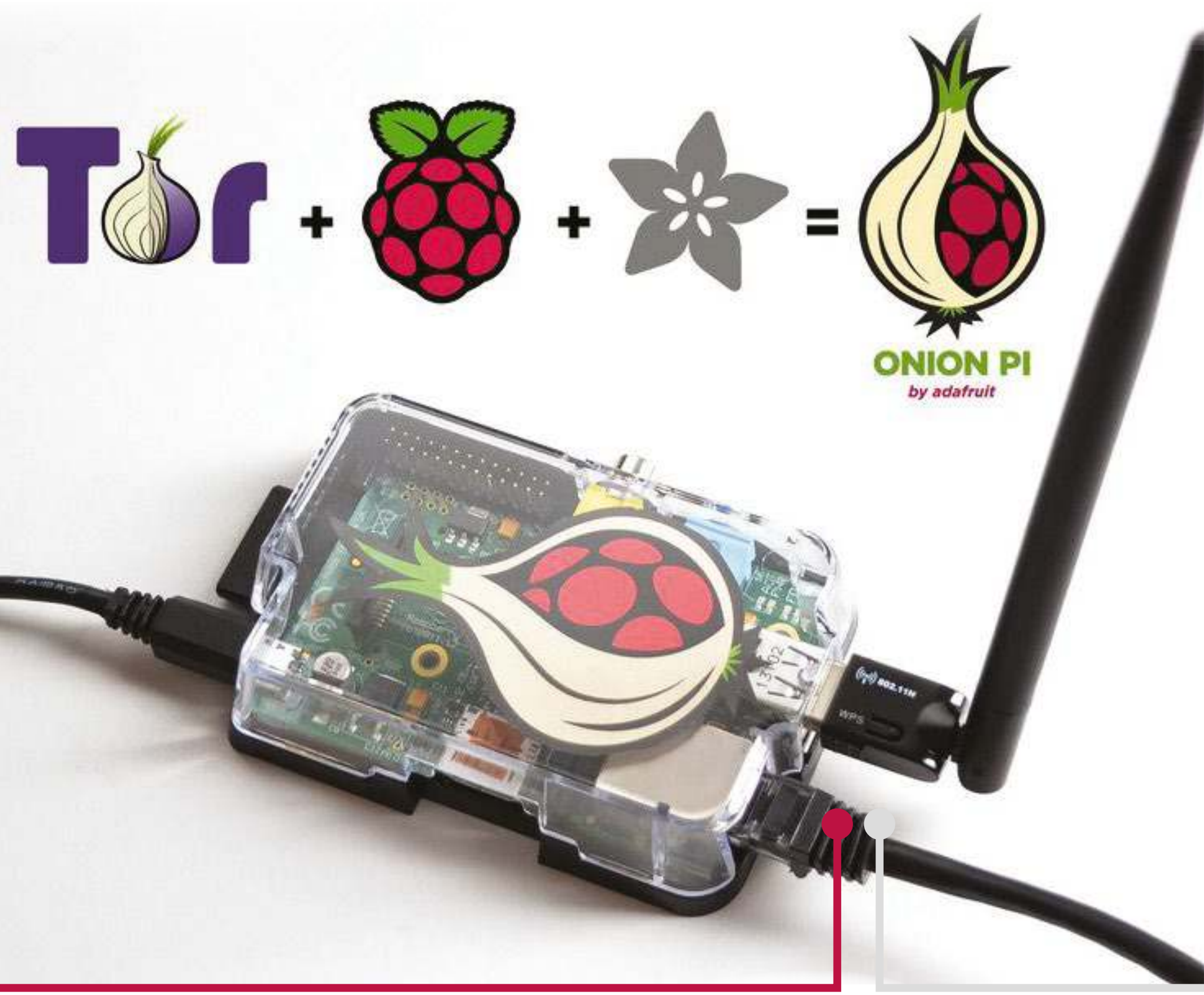
…to check the progress of the nuke. Any further attempts to log in will just take the user back to the login screen.



Left Use the 'ls' command inside the home folder to verify the home folder has been overwritten

Turn your Pi into a Tor proxy with Onion Pi

Turn your Raspberry Pi into a wireless access point to access the anonymous Tor network



In this age of ubiquitous surveillance it's harder than ever to stop hackers, advertisers and shadowy government organisations from snooping on your browsing habits.

However if you choose to connect through Tor, your connection is encrypted and passed through a number of proxies through a process known as 'onion routing'. While this does slow down your connection, it also increases your privacy, making it extremely difficult to trace your actual current location.

Follow the steps in this tutorial to turn your Pi into a wireless AP (Access Point) named Onion_Pi. Any devices connecting to Onion_Pi will do so over the Tor network. When you're done, use a service like www.whatismyip.com to see that your location has changed.

For more information about Tor visit https://www.torproject.org/about/overview.html.en.

THE PROJECT ESSENTIALS

USB Bluetooth Low-Energy device
http://bit.ly/1MtDbJC

Android/iOS device

Android/iOS beacon app

BlueZ

01 Connect to Pi and check wireless is detected

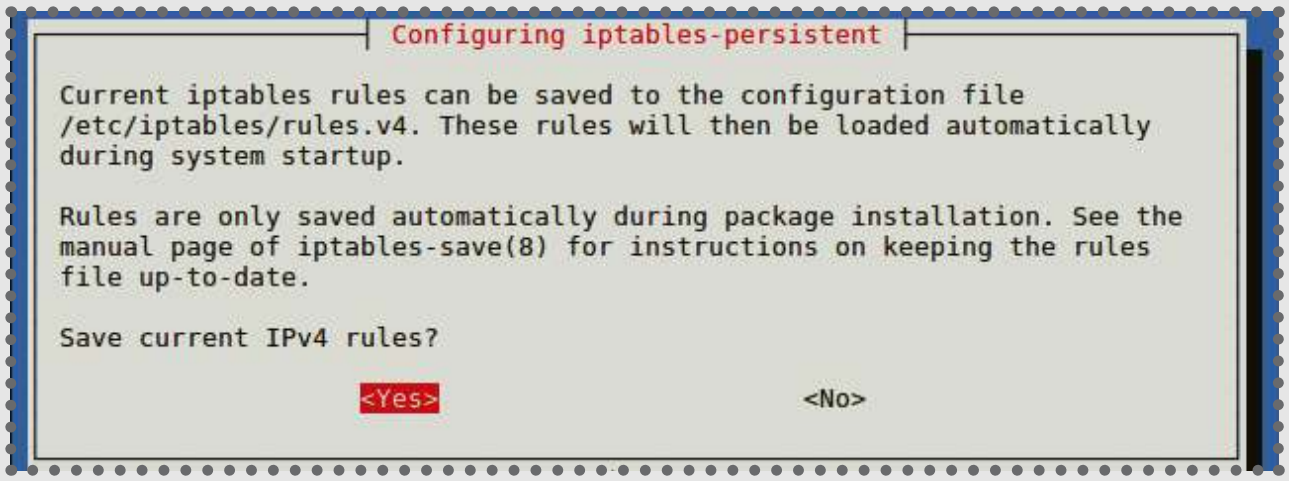
Attach your Pi to your router via the Ethernet cable, then either open Terminal on the Pi or connect to it via SSH. Run the command **sudo ifconfig -a**. You should see the text **wlan0** which shows that the wireless module is up and running.

```
          TX packets:4077 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4733748 (4.5 MiB)  TX bytes:1430164 (1.3 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:470 errors:0 dropped:0 overruns:0 frame:0
          TX packets:470 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:59771 (58.3 KiB)  TX bytes:59771 (58.3 KiB)

wlan0     Link encap:Ethernet  HWaddr b8:27:eb:59:de:10
          inet addr:192.168.42.1  Bcast:192.168.42.255  Mask:255.255.255.0
```

Left Check for 'wlan0'. Don't worry if your IP address is different. We'll change it later



Left Iptables-persistent will ask if you want to save changes. Press Return twice to agree

02 Install essential software

Run the command sudo apt-get update then **sudo apt-get install hostapd isc-dhcp-server** tor i**ptables-persistent** to install the necessary software. When you install **iptables-persistent** you'll be asked if you want to save the rules for your current configuration. Select 'Yes' both times.

03 Configure the DHCP server



Run s**udo nano /etc/dhcp/dhcpd.conf**. Find the two lines beginning 'option domain-name' and put a '#' at the start of each. Remove the '#' from the line '#authoritative'. Scroll to the end and type:

```
  subnet 192.168.42.0 netmask 255.255.255.0
{
    range 192.168.42.10 192.168.42.50;
    option broadcast-address 192.168.42.255;
    option routers 192.168.42.1;
    default-lease-time 600;
    max-lease-time 7200;
    option domain-name “local”;
```

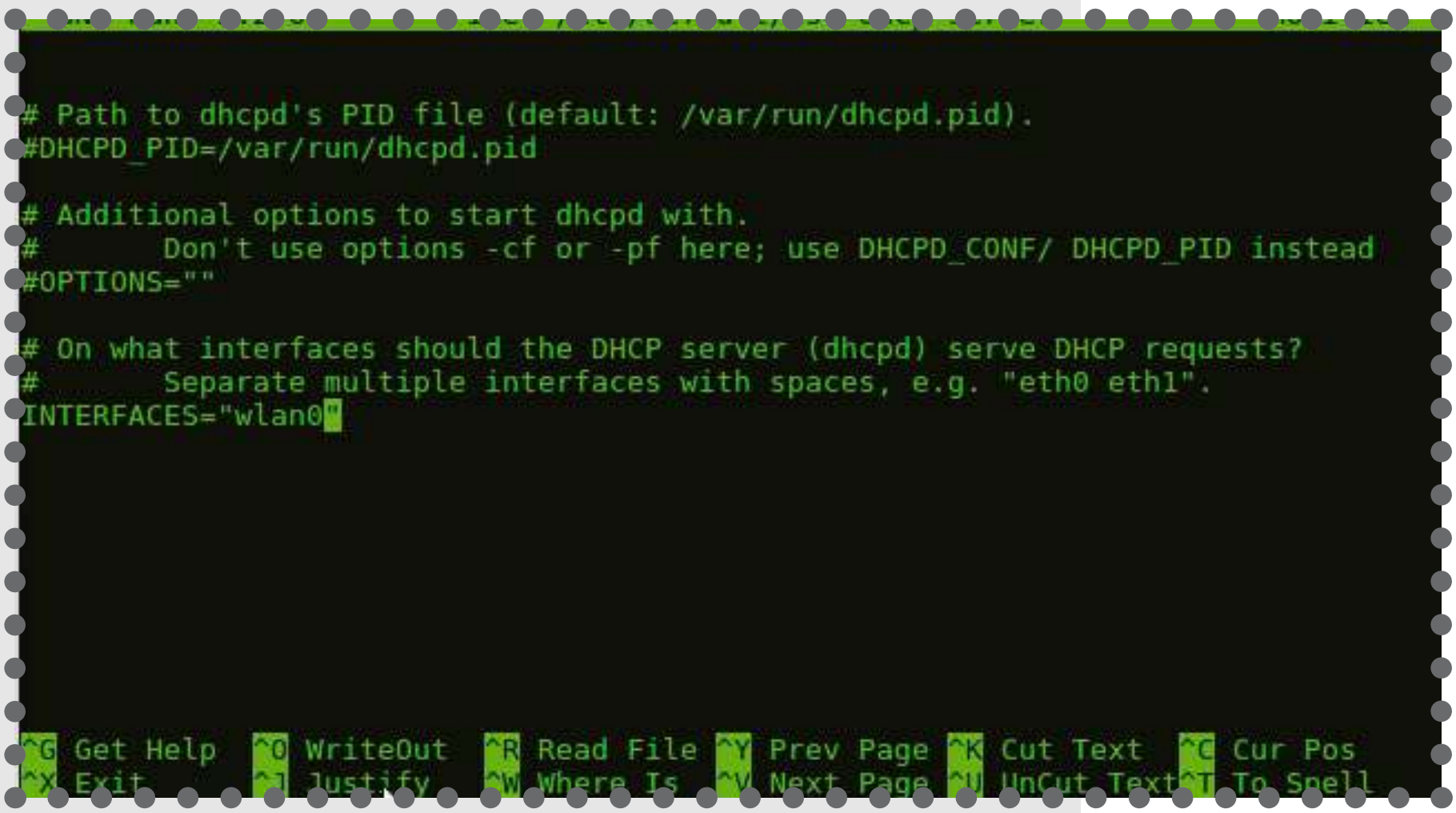
How does Tor keep you safe?

The Tor network is a group of servers or 'relays' operated by volunteers. When you start tor on the Pi, it will build a circuit of encrypted connections through relays on the network. Each relay only knows the last relay a data packet came from and where it's going, meaning it's extremely difficult to trace the data back to you. Tor also changes the circuits it uses every few minutes to make it even harder to find your machine.

```
    option domain-name-servers 8.8.8.8,
8.8.4.4;
  }
```

04 Edit interfaces

Run **sudo nano /etc/default/isc-dhcp-server**. Scroll down to the word INTERFACES="" and insert 'wlan0'. Press Ctrl + X, Y, then return to save and close. Run the commands **sudo update-rc.d hostapd enable** and **sudo update-rc.d isc-dhcp-server enable** to make sure your changes start on boot.



Above Make sure that "wlan0" is inside the speech marks

05 Set static IP

Run s**udo nano /etc/network/interfaces**. If you see the text 'auto wlan0' add a '#' at the start to comment it out. Find the line **allow-hotplug** wlan0 and delete the two lines below it. Replace them with these three

lines:

```
iface wlan0 inet static
address 192.168.42.1
netmask 255.255.255.0
```

Run **sudo ifconfig wlan0 192.168.42.1** to set your IP.

```
 GNU nano 2.2.6        File: /etc/hostapd/hostapd.conf

interface=wlan0
#driver=nl80211
ssid=Onion_Pi
country_code=US
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_group_rekey=86400
ieee80211n=1
wme_enabled=1

                [ Read 16 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C C
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T T
```

Left If you're using a Raspberry Pi 3, comment out the 'driver' line

06 Configure the Access Point

Run **sudo nano /etc/hostapd/hostapd.conf** to create a blank file. Paste in the following:

```
interface=wlan0
driver=nl80211
ssid=Onion_Pi
country_code=US
hw_mode=g
channel=6
macaddr_acl=0
```

```
  GNU nano 2.2.6               File: /etc/init.d/hostapd

#!/bin/sh

### BEGIN INIT INFO
# Provides:                hostapd
# Required-Start:          $remote_fs
# Required-Stop:           $remote_fs
# Should-Start:            $network
# Should-Stop:
# Default-Start:           2 3 4 5
# Default-Stop:            0 1 6
# Short-Description:       Advanced IEEE 802.11 management daemon
# Description:             Userspace IEEE 802.11 AP and IEEE 802.1X/WPA/W
#                          Authenticator
### END INIT INFO

PATH=/sbin:/bin:/usr/sbin:/usr/bin
DAEMON_SBIN=/usr/sbin/hostapd
DAEMON_DEFS=/etc/default/hostapd
DAEMON_CONF=/etc/hostapd/hostapd.conf

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cu
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To
```

Above Make sure you modify DAEMON_CONF="" in both init.d and hostapd.conf

```
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_group_rekey=86400
ieee80211n=1
wme_enabled=1
```

Feel free to change the passphrase from 'Raspberry' to something more complex.

07 Apply Access Point configuration

Run **sudo nano /etc/default/hostapd**. Find the line:

```
#DAEMON_CONF=""
```

Edit it so it says:

```
DAEMON_CONF=”/etc/hostapd/hostapd.conf”
```

Don’t forget to remove the **#** in front to activate it, or the line won’t work.
Repeat these same steps for hostapd with the command **sudo nano /etc/init.d/hostapd** again modifying the line **#DAEMON_CONF=””** so that it reads
DAEMON_CONF=”/etc/hostapd/hostapd.conf”

08 Configure Tor

Run **sudo nano /etc/tor/torrc to configure Tor**. Find ## **https://www.torproject.org/docs/faq#torrc** and after it paste:

```
Log notice file /var/log/tor/notices.log
VirtualAddrNetwork 10.192.0.0/10
AutomapHostsSuffixes .onion,.exit
AutomapHostsOnResolve 1
TransPort 9040
TransListenAddress 192.168.42.1
DNSPort 53
DNSListenAddress 192.168.42.1
```

Save and exit. Run **sudo update-rc.d tor enable** to make Tor start on boot.

09 Configure IP Tables

Run these commands to channel all traffic through Tor:

Tor best practices

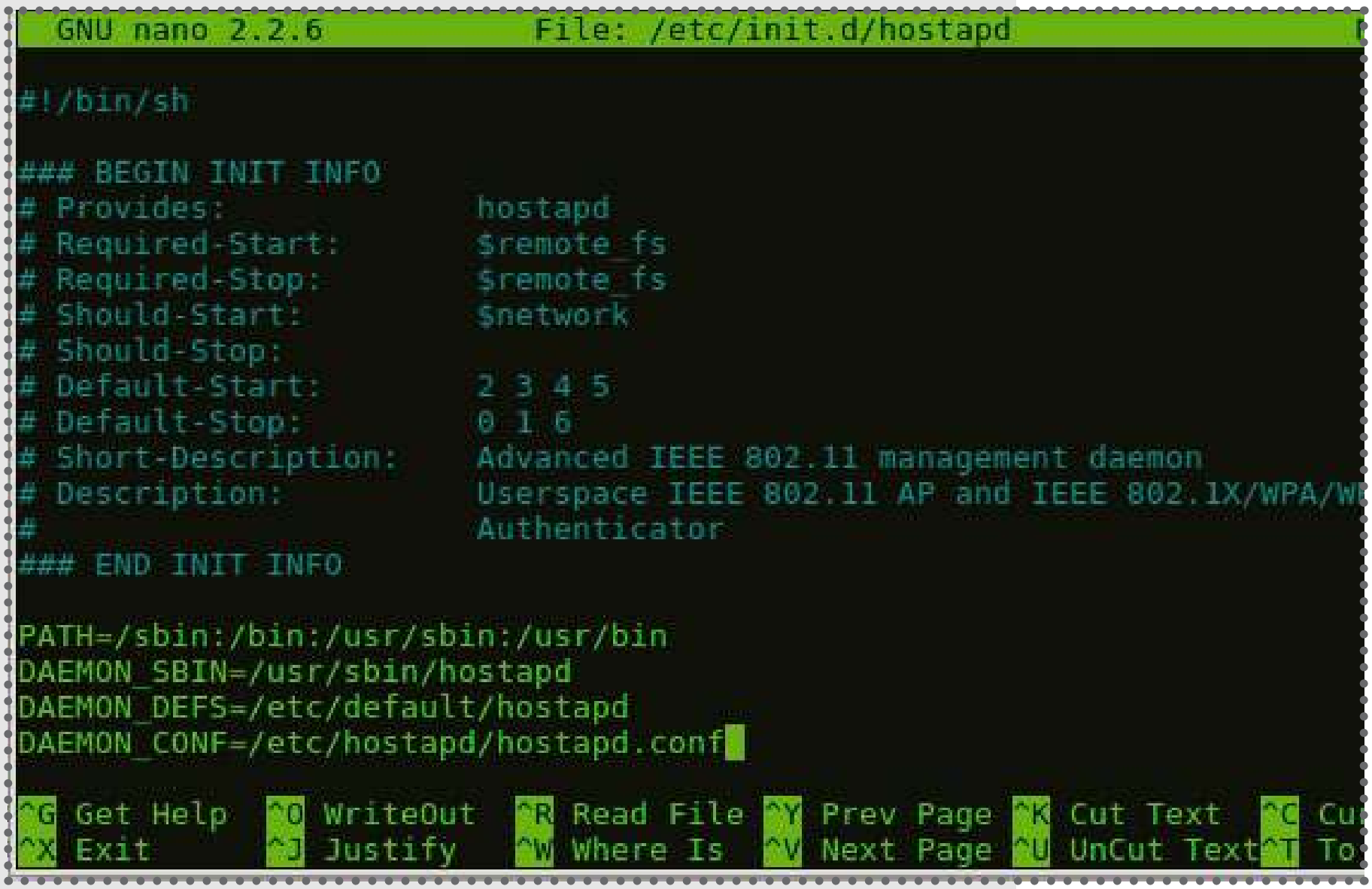
Don’t use bit torrent: this places an unfair burden on volunteers running Tor relays. Bit torrent software also often sends out your real IP address.
Use HTTPS versions of websites: this encrypts your connection between your Tor exit node and your target **website e.g https://gmail.com**
Don’t use browser plugins: this makes it easy to fingerprint your browser.
Don’t open online documents: PDF and .doc files can contain code that might give away your real location.

```
  sudo iptables -t nat -A PREROUTING -i
wlan0 -p tcp --dport 22 -j REDIRECT --to-
ports 22
  sudo iptables -t nat -A PREROUTING -i
wlan0 -p udp --dport 53 -j REDIRECT --to-
ports 53
  sudo iptables -t nat -A PREROUTING -i
wlan0 -p tcp --syn -j REDIRECT --to-ports
9040
```

Next, make your changes permanent:

```
  sudo sh -c “iptables-save >  /etc/
iptables/rules.v4”
```

Reboot your Pi when done.



Below Use the command sudo iptables -t nat -L when you’re done to check your configuration

Encrypt your Raspberry Pi home folder

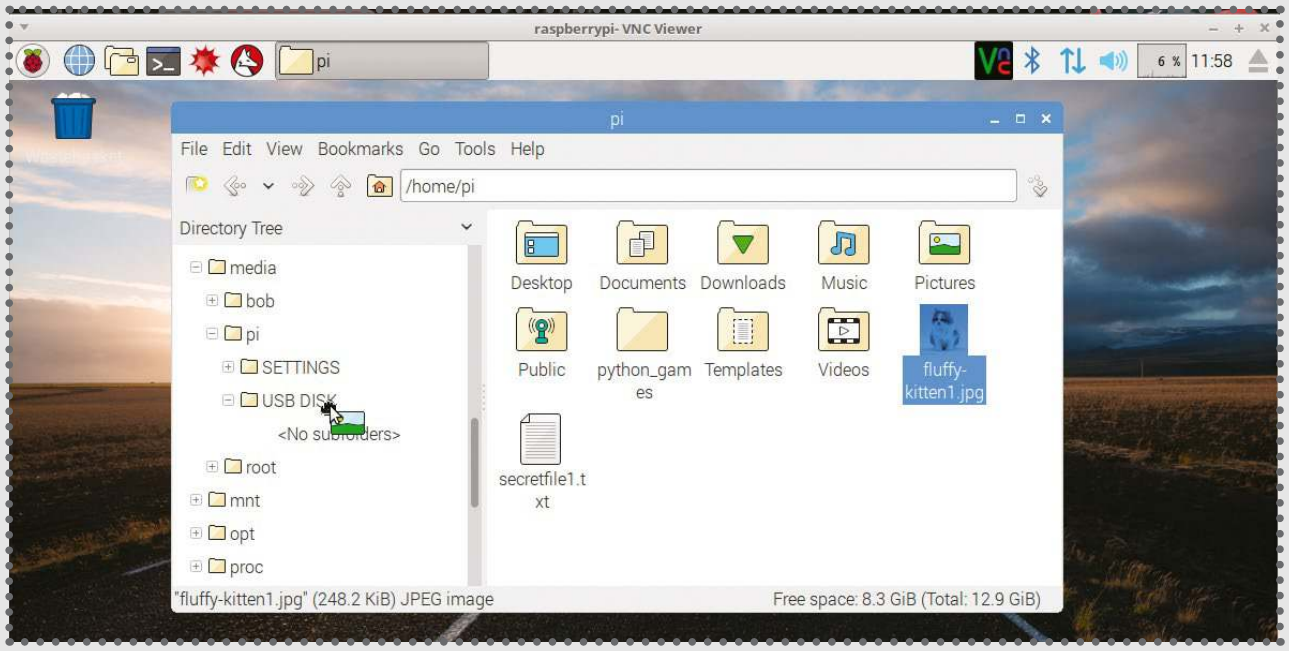
Easily keep your data safe with military-grade encryption on your Pi



THE PROJECT ESSENTIALS

This tutorial is compatible with all models of Pi running Raspbian Jessie

With the advent of the dazzling Pixel desktop on the Pi, with its glitzy icons and crisp windows, some readers may be considering using a Pi as their home computer. This however is not especially secure given that the Pi automatically logs in the default user without requiring a password. Even if the password is required, anyone can mount the SD card to view your documents, pictures, videos and other data.



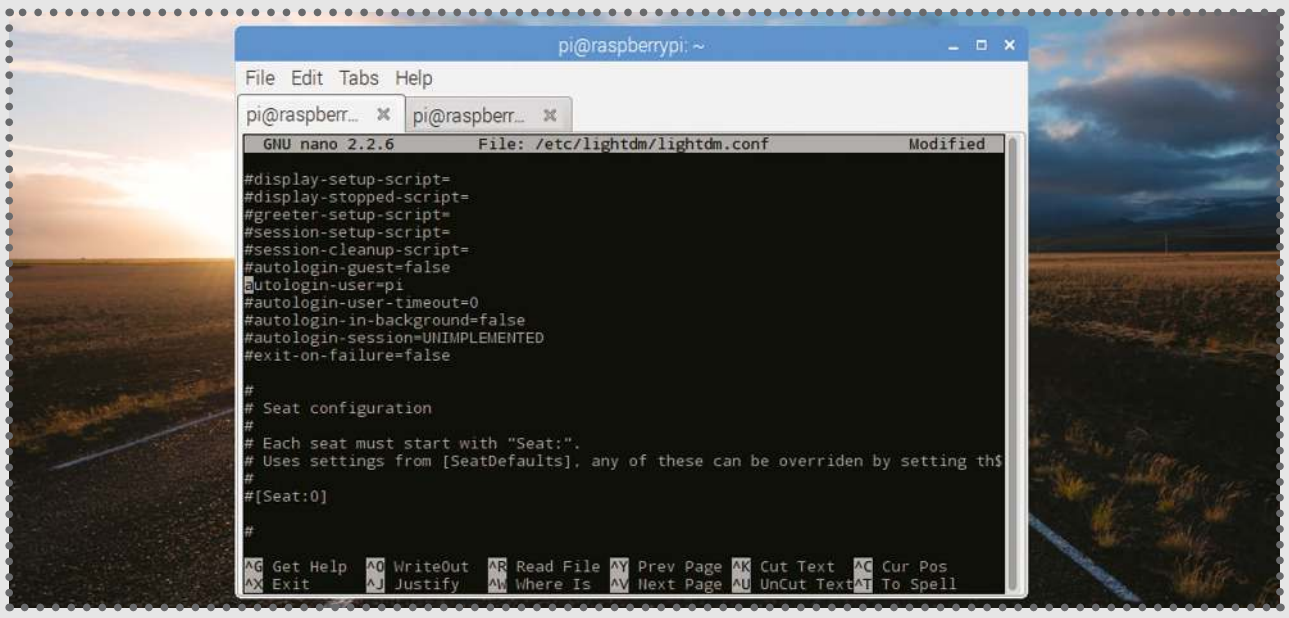
01 Back up your data

If you have not done so already, log onto your Pi and transfer the contents of your Desktop, Documents, Pictures etc onto an external medium like a USB stick. This will make transferring your data to your encrypted home folder much easier. Use Ctrl+H to show hidden folders if you want to back up your application settings such as your browser bookmarks.

02 Enable login screen

If you have not already done so, open Terminal on your Pi or connect via SSH and run the command:

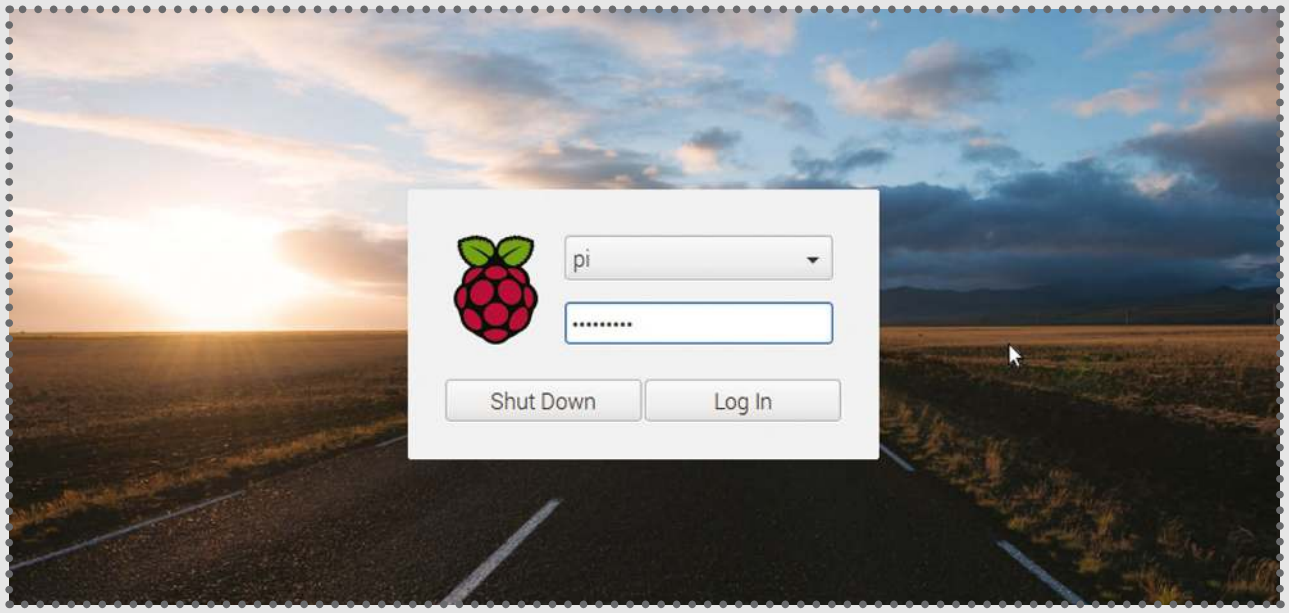
```
sudo nano /etc/lightdm/lightdm.conf
```



to open your login options. Scroll down to the line **autologin-user=pi** and put a hash (#) at the start to comment it out. **Press Ctrl+X**, then Y, and then Return to save and exit. Remember the default password is 'raspberry'.

03 Reboot Pi and log in

Reboot the Raspberry Pi and log in via the login screen. The username pi should already be selected and the password should be 'raspberry'. Moving forward we recommend that you work on the Pi's desktop directly as you will need to switch between users, which is difficult over SSH. If you do not have a monitor for your Pi, try connecting via VNC as the 'pi' user.

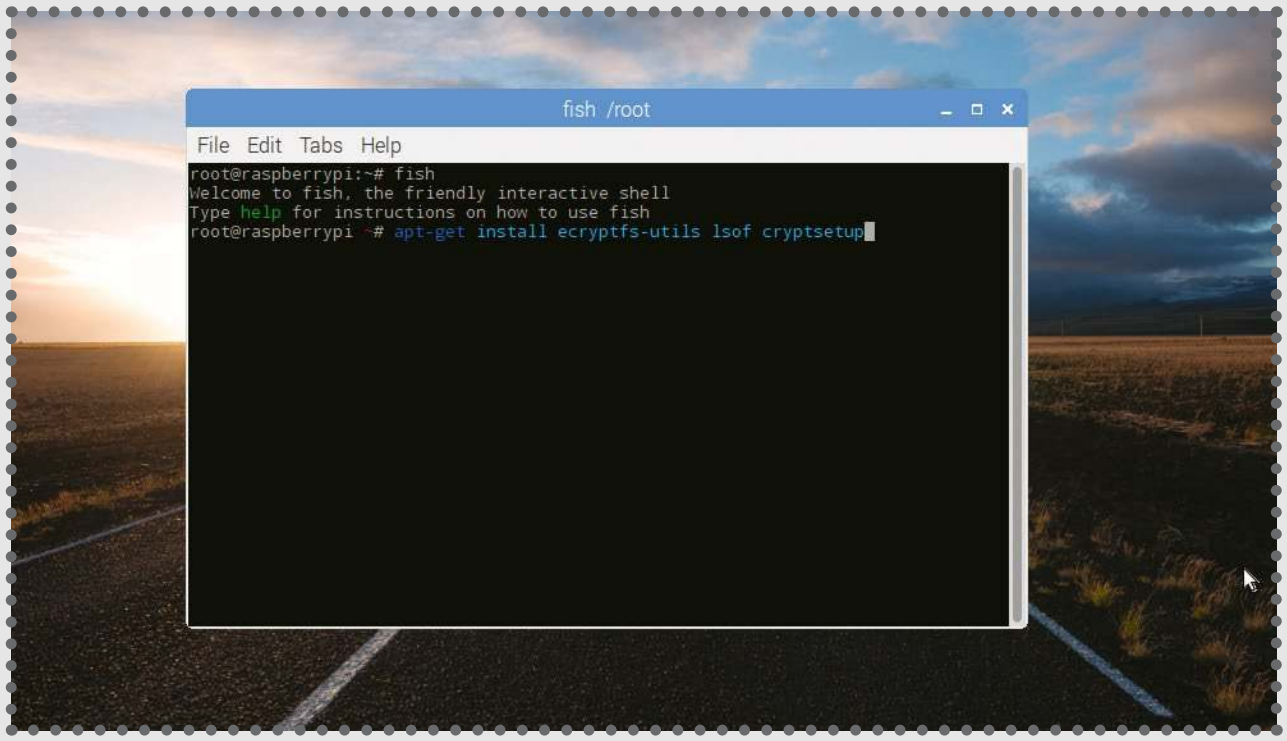


04 Install eCryptfs and related files

Open Terminal on the Pi and run the command:

```
  sudo apt-get install ecryptfs-utils lsof
cryptsetup
```

You will need to press Y to confirm that you do indeed



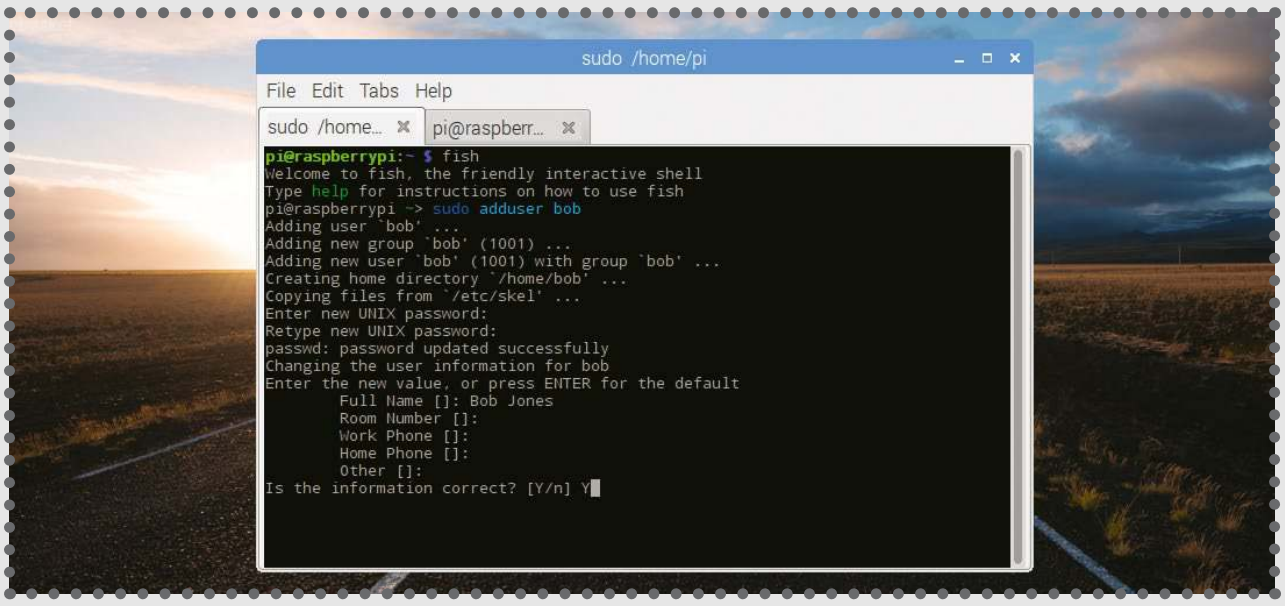
want to install the software. This will allow you to encrypt the home directory and access it each time you log in.

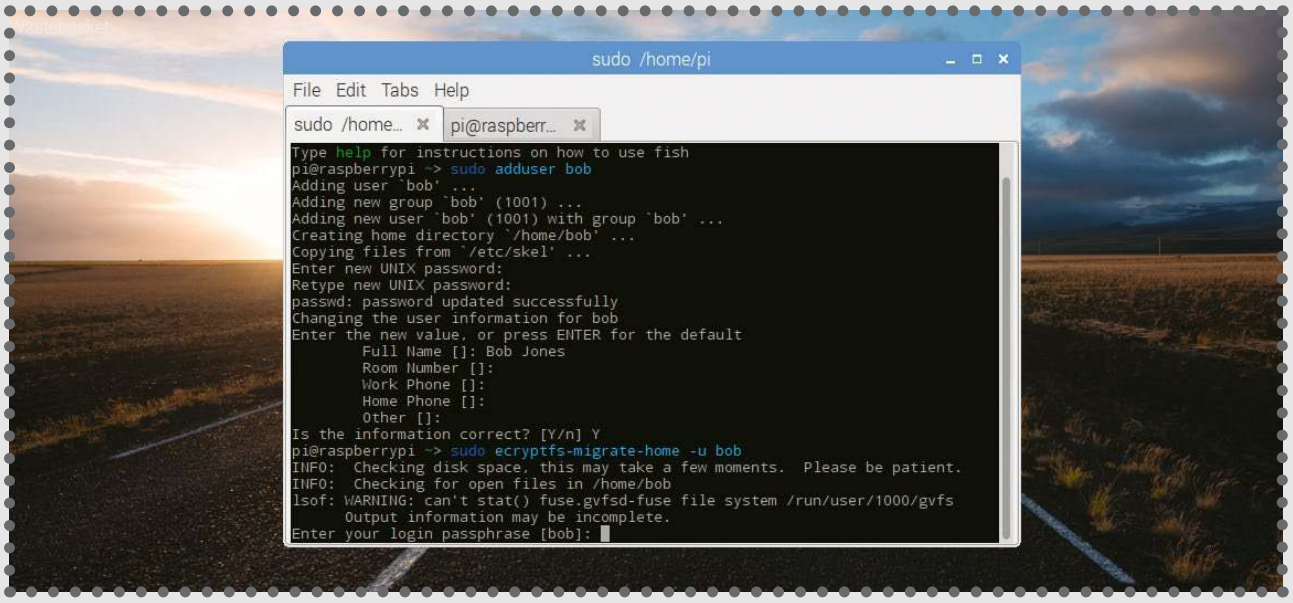
05 Create new user

If you are serious about wanting to use the Pi, you most probably will want an account in your own name in any case. Use the command:

```
sudo adduser (username)
```

e.g sudo adduser bob to create your account. Type your password twice, then press Return to accept default values for the other options such as location.



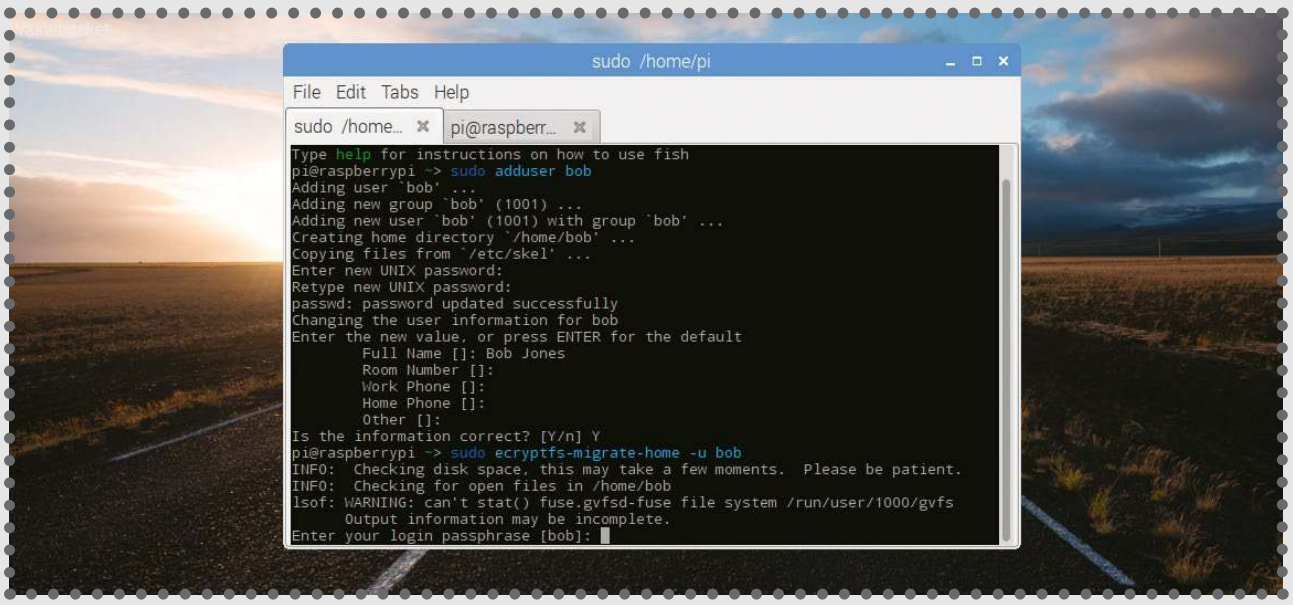


06 Encrypt new user's home directory

The eCryptfs software comes with a handy built-in utility to encrypt existing home folders. Simply run the command:

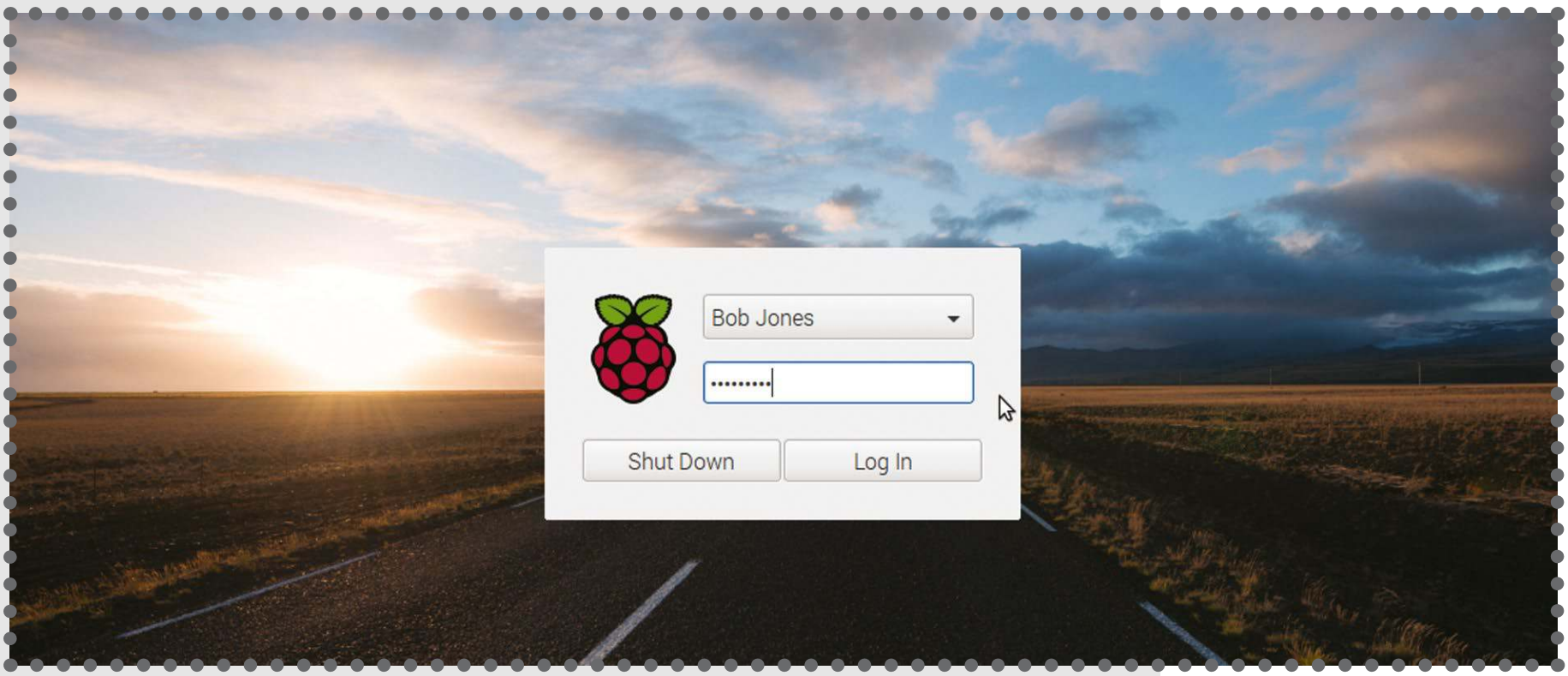
```
sudo ecryptfs-migrate-home -u bob
```

where 'bob' is your new username. You'll be asked to enter a login passphrase twice (this can be the same as the password you just created). Make sure to read the Important Notes section once this is done. Do not reboot at this stage.



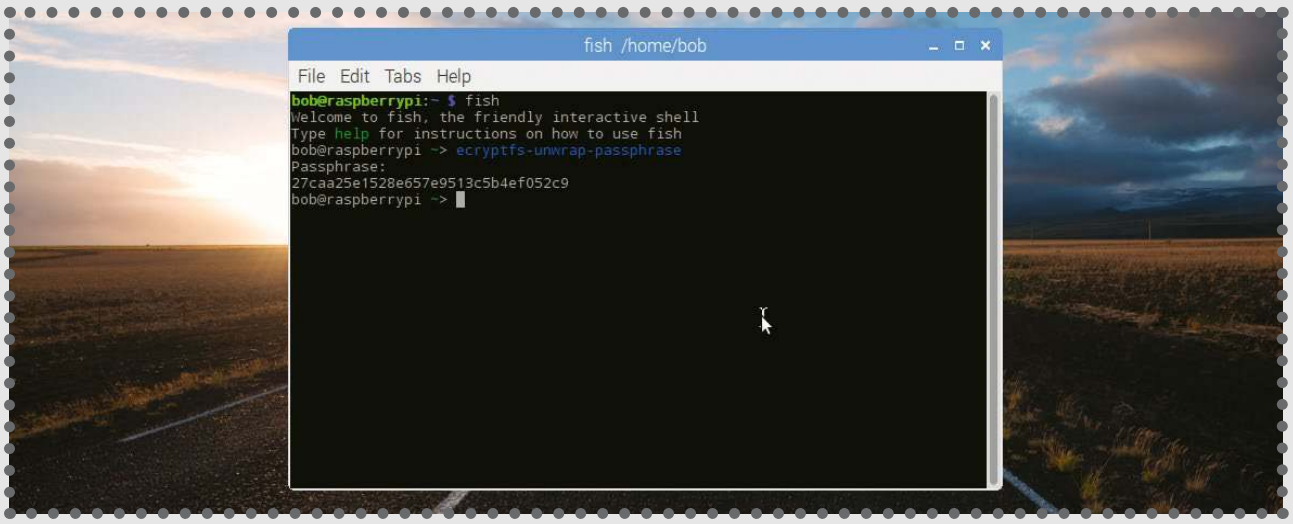
07 Log into new encrypted user account

Use Menu>Shutdown>Logout to log out of the Pi user. In the dropdown menu select your new username e.g 'bob' and log in. Before bringing any data into this new account, open Terminal and simply type the command mount. This will show a flurry of information; you should see a reference to 'ecryptfs', which shows the encryption has been successful.



08 Back up your passphrase

If you're unable to log in because of a system problem or you want to move your data to a new computer, by default your files won't be accessible. Fortunately



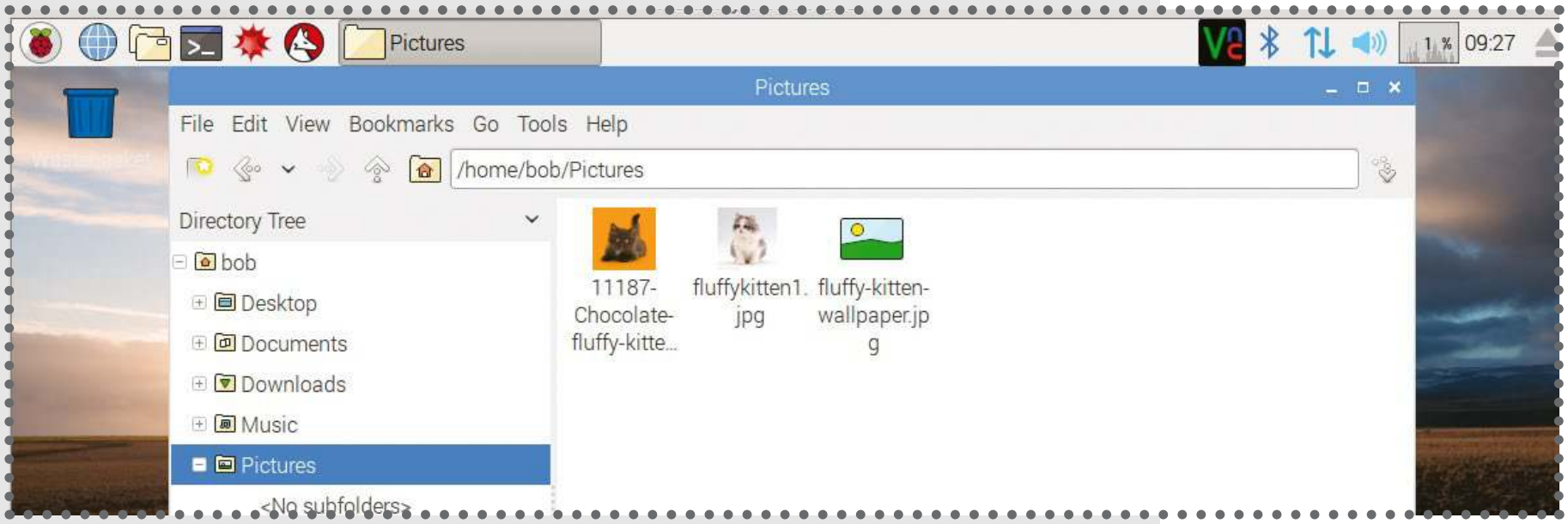
there's an eCryptfs utility that can generate your mount passphrase, which can be used to access your files from another device. Open Terminal and run the command

```
ecryptfs-unwrap-passphrase
```

to view the passphrase and write it down. Store the passphrase in a safe place.

09 Migrate your data

Use Menu>Shutdown>Reboot to restart your Pi and then log back in as the new user. At this stage you should connect your external drive and begin copying your documents and data back to the right places. You'll see that a new home folder has been created in /home, e.g /home/bob. No other users on the Pi will be able to access the files inside your home folder.



10 Give your new user admin privileges

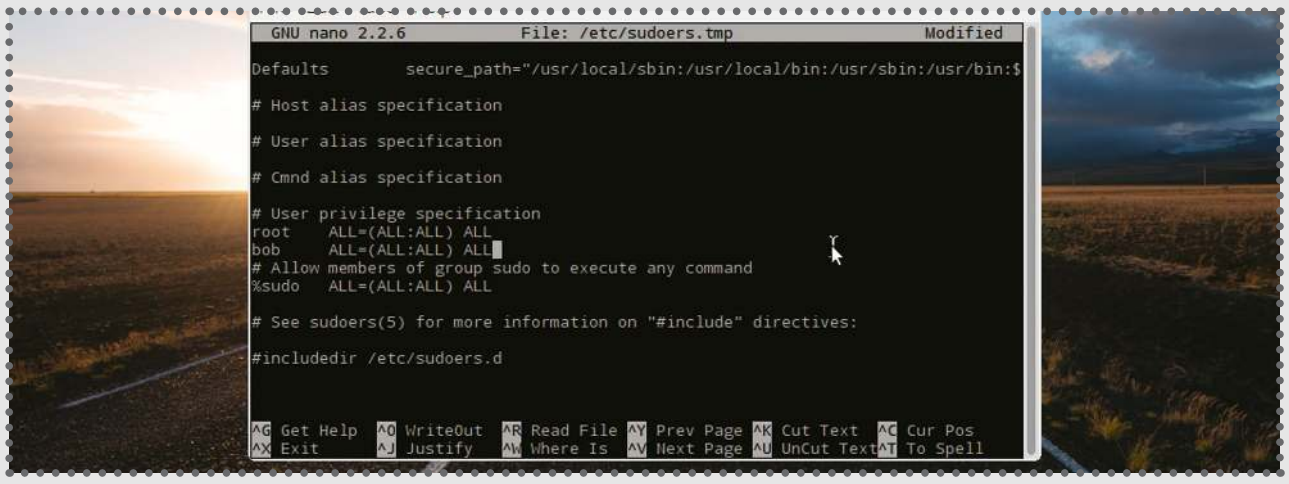
Log out of your new user for now and back into 'pi'. Open Terminal and run the command:

```
sudo visudo
```

Scroll down to the line reading root **ALL=(ALL:ALL) ALL** and on a new line immediately after this add:

```
bob     ALL=(ALL:ALL) ALL
```

where 'bob' is your new username. **Press Ctrl+X**, Y, then Return to save and exit. Restart the Pi again to effect your changes.

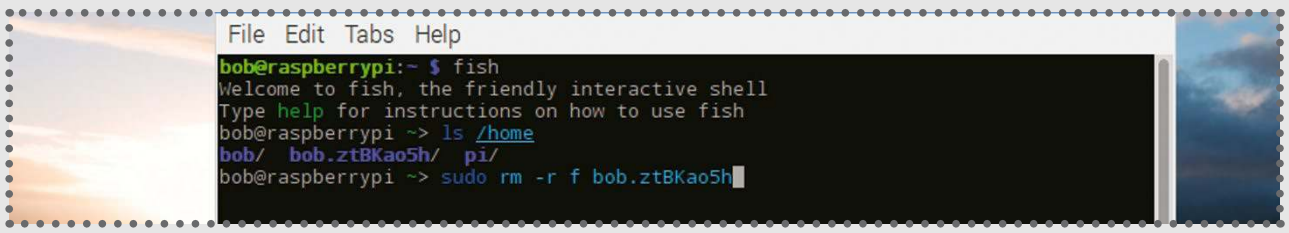


11 Remove backup home folder

When encrypting your new home directory, the eCryptfs software places a backup in the home folder in case anything goes wrong. As this was a new account, the folder is empty, but to keep things simple, open Terminal and run the command :

```
ls /home
```

to find out its exact name, e.g bob.zyxxc, then run the



Choosing a good password

Even if your home folder is encrypted it can be brute-forced by another computer trying various passwords until it hits on yours. Visit the website https://howsecureismypassword.net/ to see how long it would take to crack your own password. If the result is anything less than a century or so, consider using a stronger password. The DiceWare website (http://world.std.com/~reinhold/diceware.html) provides a quick and easy way to generate strong and easy to remember passwords through randomly selecting several dictionary words.

command:

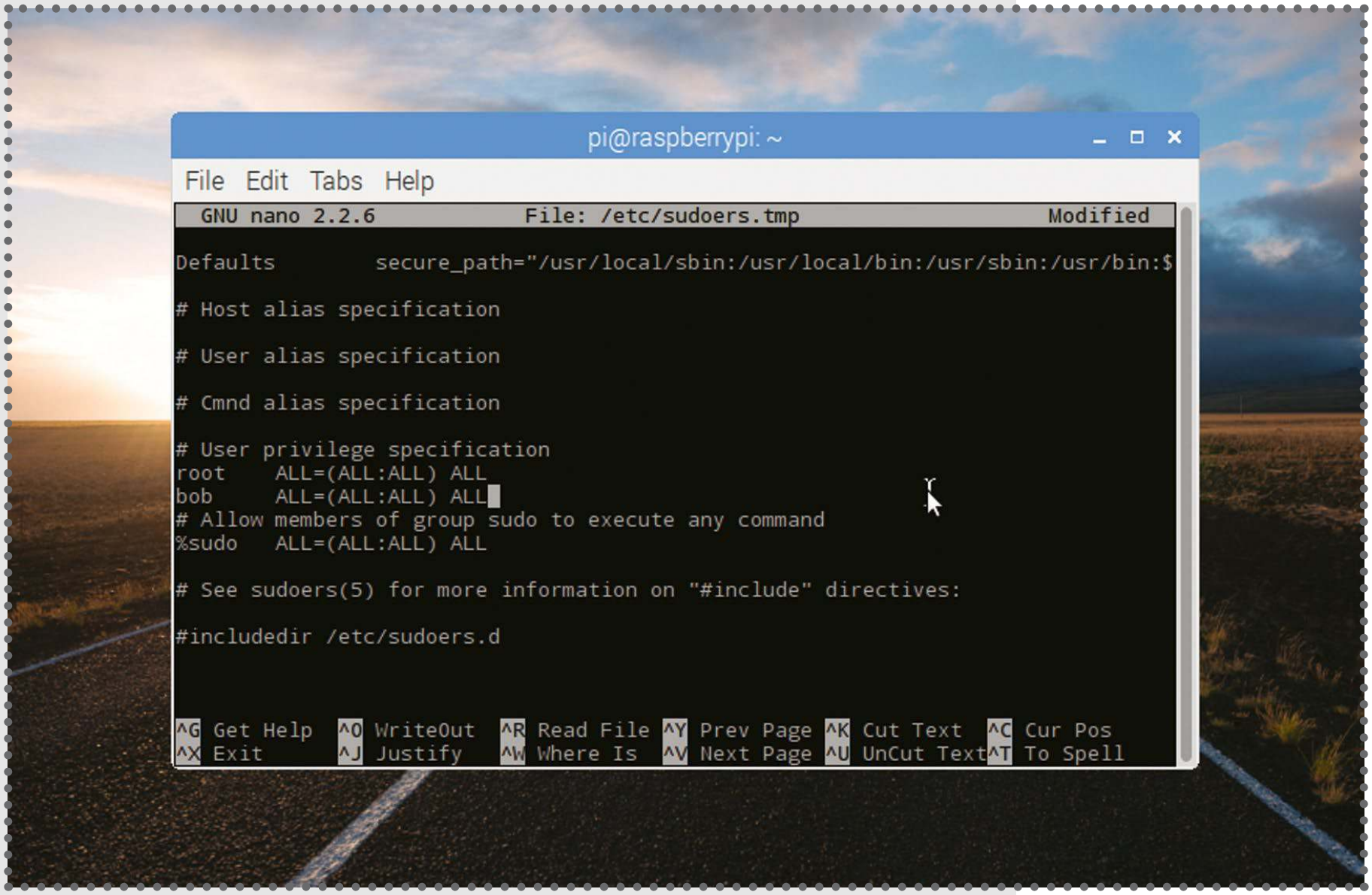
```
sudo rm -r -f <directoryname>
```

to remove it.

12 Delete data in Pi home folder

If you had personal data already in the existing 'pi' folder, it is still unencrypted and can be accessed by anyone who can obtain your Pi's MicroSD card. Log onto the 'pi' user and use the shred command to securely delete any files you want e.g

```
shred -zu bank-statement.pdf
```



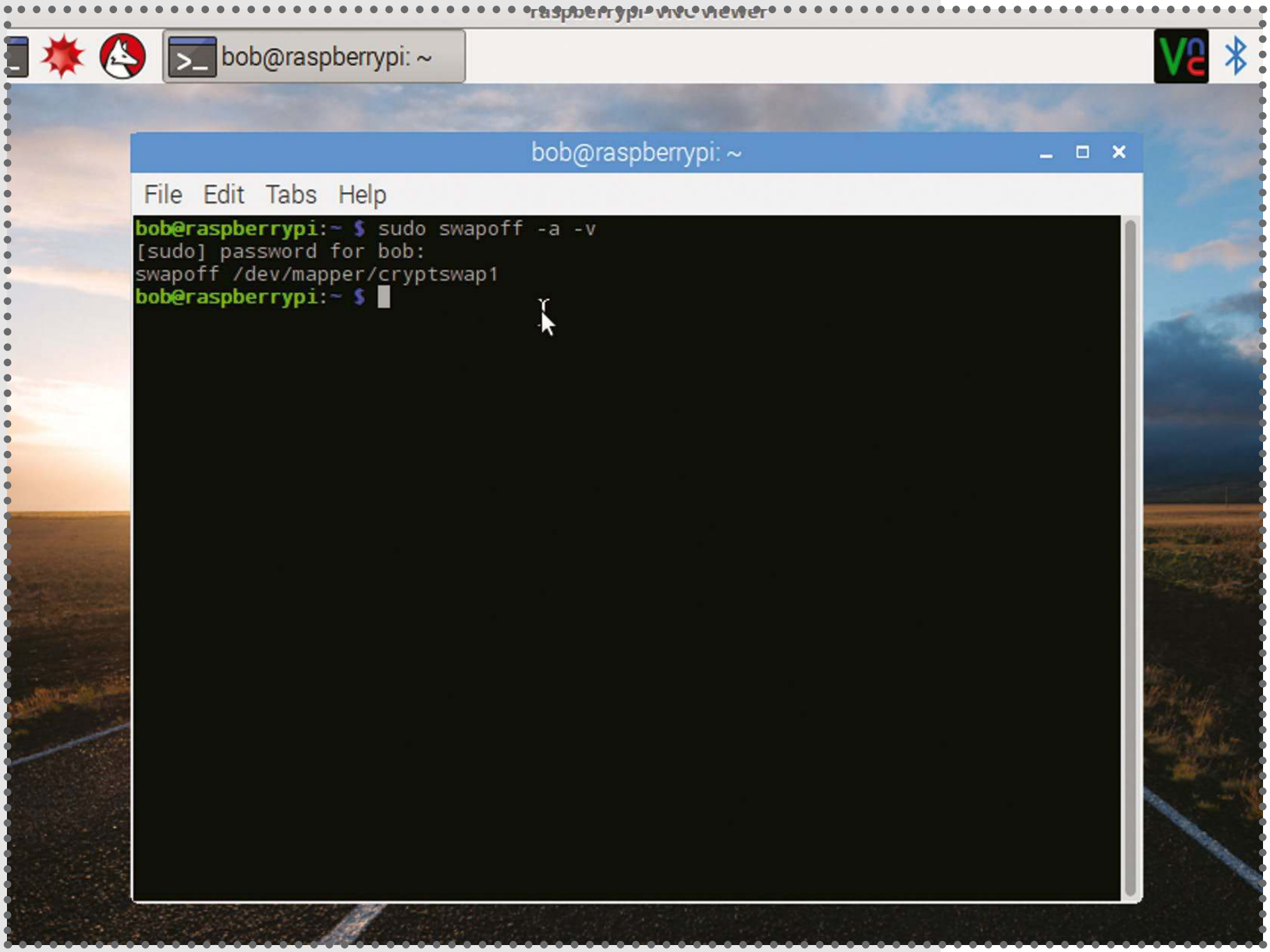
You can use

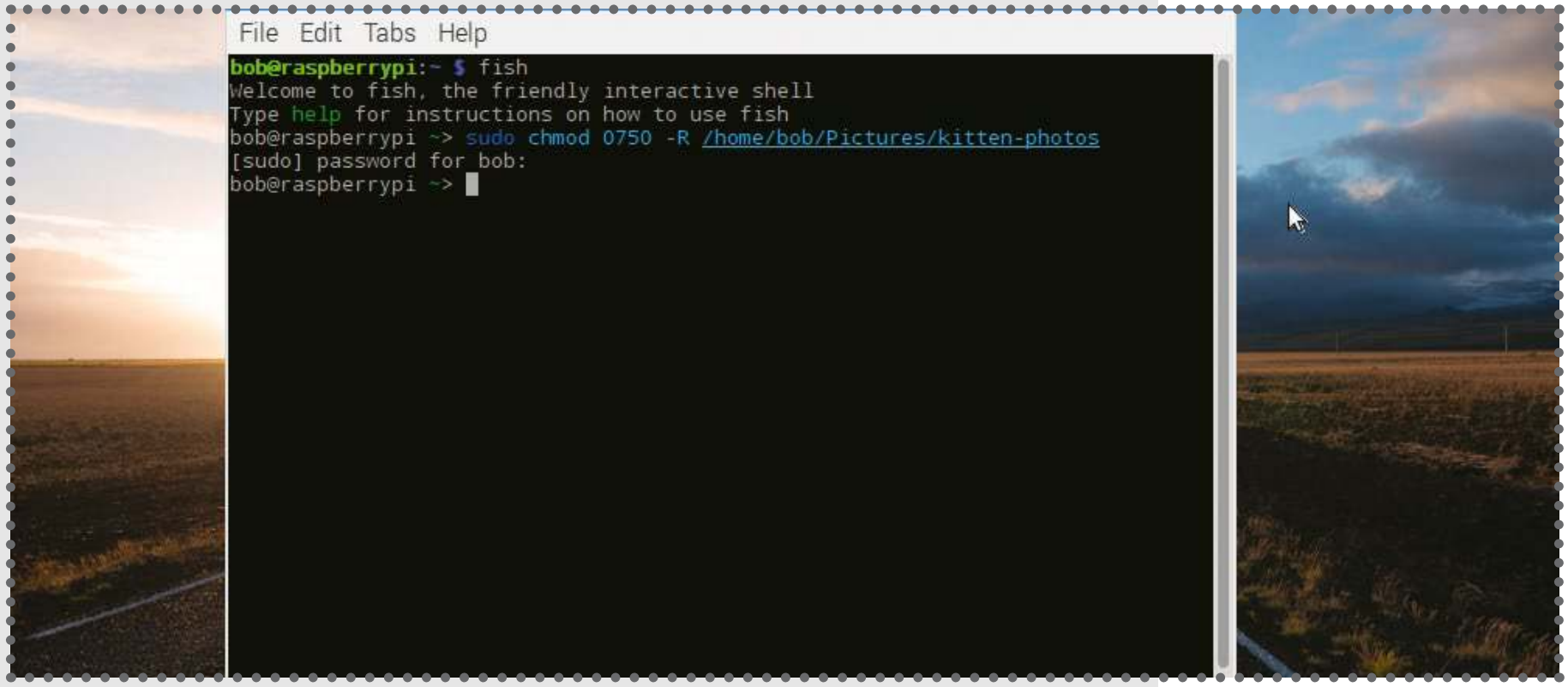
```
sudo rm -r -f <directoryname>
```

to erase entire folders but do not do this with any of the main folders in your 'pi' home folder e.g Desktop.

13 Disable swap space

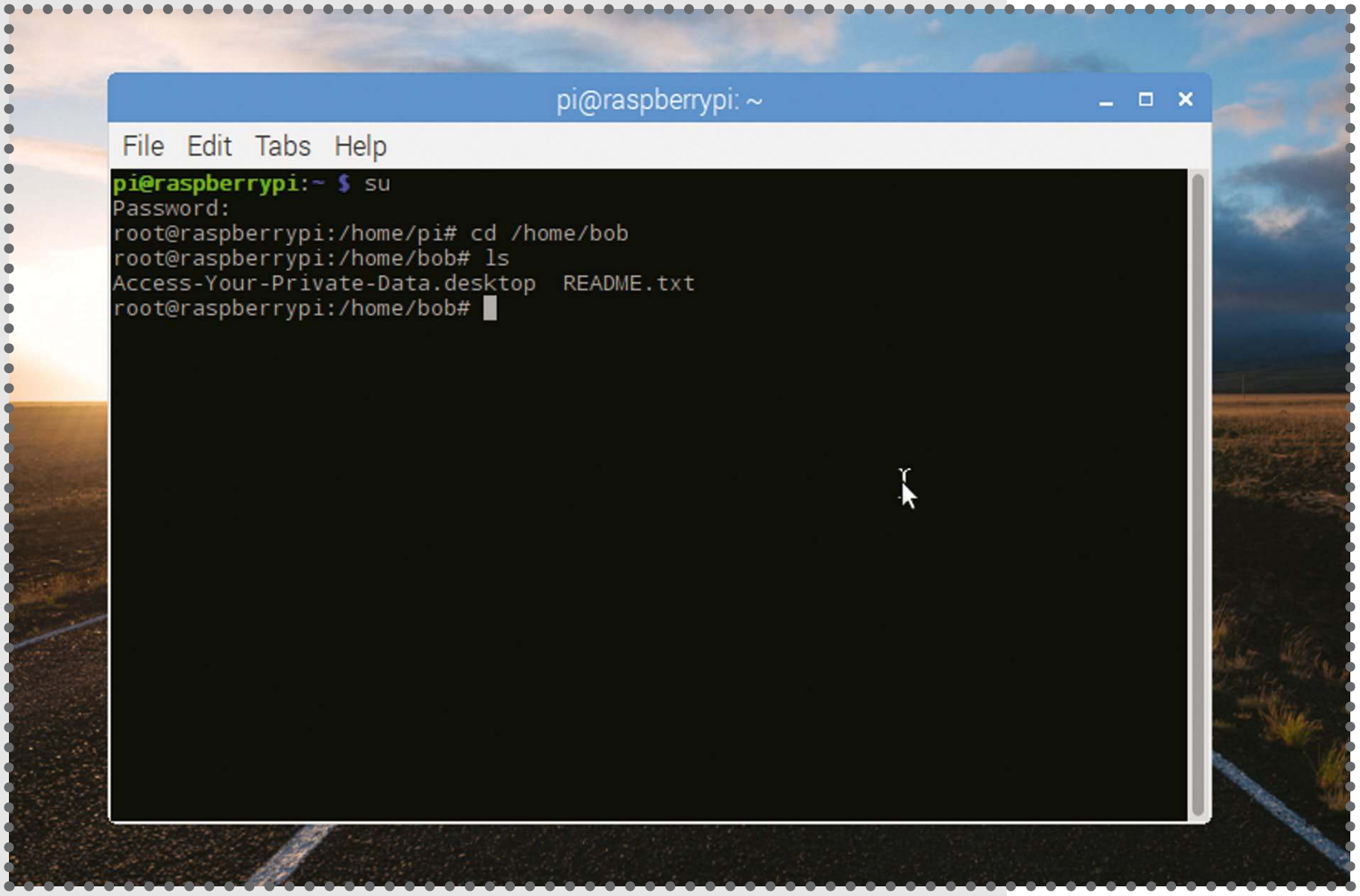
There is a portion of the Pi's SD card (located in /var/swap) that is used in a way similar to RAM when the Pi is low on resources. For more modern Pi models this is very rarely used and can present a security risk as your data may be written there unencrypted. Log in to your new user account, open Terminal and run:






```
sudo swapoff -a -v
```

You should see a confirmation that it has been disabled.



14 Fix permissions errors

Depending on the method you used to back up and transfer your data from your previous account, there may be permissions issues with certain files and folders (Usually you will see a padlock on files you're unable to edit). You can make sure your new account is the owner of all files within folders with the command:

```
sudo chmod 0750 -R foldername
```

e.g sudo chmod 0750 -R /home/bob/Pictures, where 'bob' is your new username.

15 Double check your files are safe

This step is optional. Reboot the Pi and log into the 'pi' user. Open Terminal and if you have not previously done so, enter
the command:

```
sudo passwd root
```

to set a password for the root user. Next enter the command su to switch into root. Run:

```
cd /home/bob
```

(where 'bob' is your new username) and then use **ls** to show the contents of the directory. There's now only a shortcut and a 'ReadME' file saying your data is protected.

Monitor servers with a Raspberry Pi

Raspberry Pis are a great platform for any kind of monitoring project. This issue, learn how you can keep an eye on your servers with a Raspberry Pi

Even with a small footprint, Raspberry Pis can still provide a powerful set of computing resources. This is the perfect combination for projects that are meant to monitor something. We will look at how to code up a system to be able to keep an eye on an important server that you need information about. There are several ways to do this. Since we have limited space here, don't forget to check out other options in case something else might work better in your situation. As with other articles in this series, we will be using Python to write the code that handles monitoring the external server.

A properly hardened server should only allow encrypted connections over SSH from outside machines. There are several options available for modules that you could install. A pure Python module that is fairly complete is Paramiko. This is the package that we will use for this article. Install it on your Pi with the command:

```
sudo apt-get install python3-paramiko
```

If you are using Python 2.X, you can change the

module name to be 'python-paramiko'. This command also installs the Python modules that support cryptographic operations. Once you have the module installed, you can get your code prepared with the following code:

```
import paramiko
client = paramiko.SSHClient()
client.load_system_host_keys()
client.connect('192.168.0.10',  username='pi', password='raspberry')
```

On some occasions you may find that this piece of code may or may not work. It all depends on whether you have already connected to the external server with SSH before. If you have, then an identifier named the host key should have been saved in the file ~/.ssh/known_hosts. If so, then it should be in the set of keys imported by the load keys method. If you have never made an SSH connection to the server in question, then it will not be in the known hosts file and you will get an exception. You set a callback function to deal with this situation manually. If you are only going to be connecting to known machines, there is a shortcut you can use instead. Paramiko includes a helper method that will automatically add new hosts to the known hosts list for you. You can turn this functionality on with the following line of code:

```
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

You need to add this line somewhere before you try and make your first connection. If you want to minimise the chances of security issues, Paramiko gives you helper methods to manage the host key file. This code accepts a host key and then saves it off to the standard file:

```
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
client.connect('192.168.0.10', username='pi', password='raspberry')
client.save_host_keys('/home/pi/.ssh/known_hosts')
```

You can reload this host key file in regular usage with the following code segment:

```
client.load_host_keys('/home/pi/.ssh/known_hosts')
client.connect('192.168.0.10', username='pi', password='raspberry')
```

So, now we have an open SSH connection to the server you wish to monitor. What can you do with this? Since you want to monitor the health of the remote machine, you need to execute commands on this remote machine and get the results back to analyse and/or display them. The simplest way to do this is the exec_command() method. This code snippet shows how you can get the current uptime of the server:

```
stdin, stdout, stderr = client.exec_command('uptime')
```

```
curr_uptime = stdout.readlines()
print(curr_uptime)
```

The output you get will look like this:

```
[‘ 21:55:43 up 13 days, 22:34,  0 users, load average: 0.00, 0.00, 0.00\n’]
```

In the original code snippet, the method exec_command() returns a tuple of values for the standard input stream (stdin), the standard output stream (stdout) and the standard error stream (stderr). When you call this method, a new Channel is opened and the three standard IO streams are attached. You can then read output from the executed command from the stdout and stderr streams. The three streams behave much like standard Python file objects. This means when you use the readlines() method, you will not be able to go back and reread that data. This means that you should always save off this incoming data to a variable.

This works for simple commands, but what if you need a more complicated command that requires your interaction? An example is any command that requires sudo access to run. Your user account would need to have sudo permissions, and then you’d need to be able to type in your password when asked. Luckily, you have the stdin stream from the exec_command() method that lets you send in the required input. This code shows how to get the output from the dmesg command:

```
stdin, stdout, stderr = client.exec_command(‘sudo dmesg’, get_pty=True)
```

“A pure Python module that is fairly complete is Paramiko”

```
stdin.write('raspberry\n')
stdin.flush()
dmesg_data = stdout.read().splitlines()
```

With the exec_command() method, we introduced a new parameter named get_pty. By default, this parameter is False, but you may need to set it to True for some remote machines. The write() method for the stdin stream takes your password, to be handed in to the sudo process. You need to include the newline character at the end of your password to represent you hitting the Enter key on the keyboard. Once this has been executed, you need to call the flush() method to make sure that this data is pushed to through the SSH connection to the remote machine. You can then read the returned results from the dmesg command. In the above example, we just used the read() method, and then used the splitlines() method to end up with a list of the returned lines. If you have something specific you are looking for, you can loop through the returned lines. For example, the following code looks for all of the lines that are messages from the systemd process.

```
for line in dmesg_data:
    if line.split(':')[0] == 'systemd':
        print(line)
```

This can work for any system software that might be running on your monitored server.

The last thing you might want to be able to do with the server is to move files on and off the remote machine. It might be that you need to pull log files off the system for further analysis. The beginning of working

with files is exactly the same as above, up to the point that you have a valid client object ready that has opened a connection to the remote machine. The following code shows how to get a log file off of the server.

```
sftp = client.open_sftp()
sftp.get('localcopy.log', 'serverside.log')
sftp.close()
```

This code opens an SFTP connection to the server and calls the get method. This gets the remote file, named serverside.log, and saves it to the local file localcopy.log in the current working directory on your local Raspberry Pi. Once you have finished with all of your file IO, you need to use the close() method to clean up the connection details.

Putting this together, the following code shows how you can check the load average on the server and print out a warning if it gets higher than a certain value.

```
import paramiko
client = paramiko.SSHClient()
client.set_missing_host_key_
policy(paramiko.AutoAddPolicy())
client.connect('192.168.0.10', username='pi',
password='raspberry')
stdin,stdout,stderr = client.exec_
command('cat /proc/loadavg')
data1 = stdout.readlines()
data2 = data1[0].split()
if float(data2[0]) >= 5.0:
  print('HELP! TOO MUCH WORK!')
```

Why Python?

It's the official language of the Raspberry Pi. Read the docs at **python.org/doc**

```
client.close()
```

This will scream at the user, in all caps, when the one-minute load on the remote machine gets to at least 5.0.

For very secure systems, you may not be allowed to connect using a username and password combination. If this is the case, you will need to use an SSH key to authenticate yourself with the remote machine. If you have the keys already set up correctly within the directory ~/.ssh, you can simply call the connection() method with no password. Paramiko will search in the standard places for the relevant key files. If you just have a file containing the relevant key information, you can hand this into the connection() method. This example shows how to use the key file that you placed in the current working directory.

```
client.connect('192.168.0.10', username='pi', key_filename='mykey.pub')
```

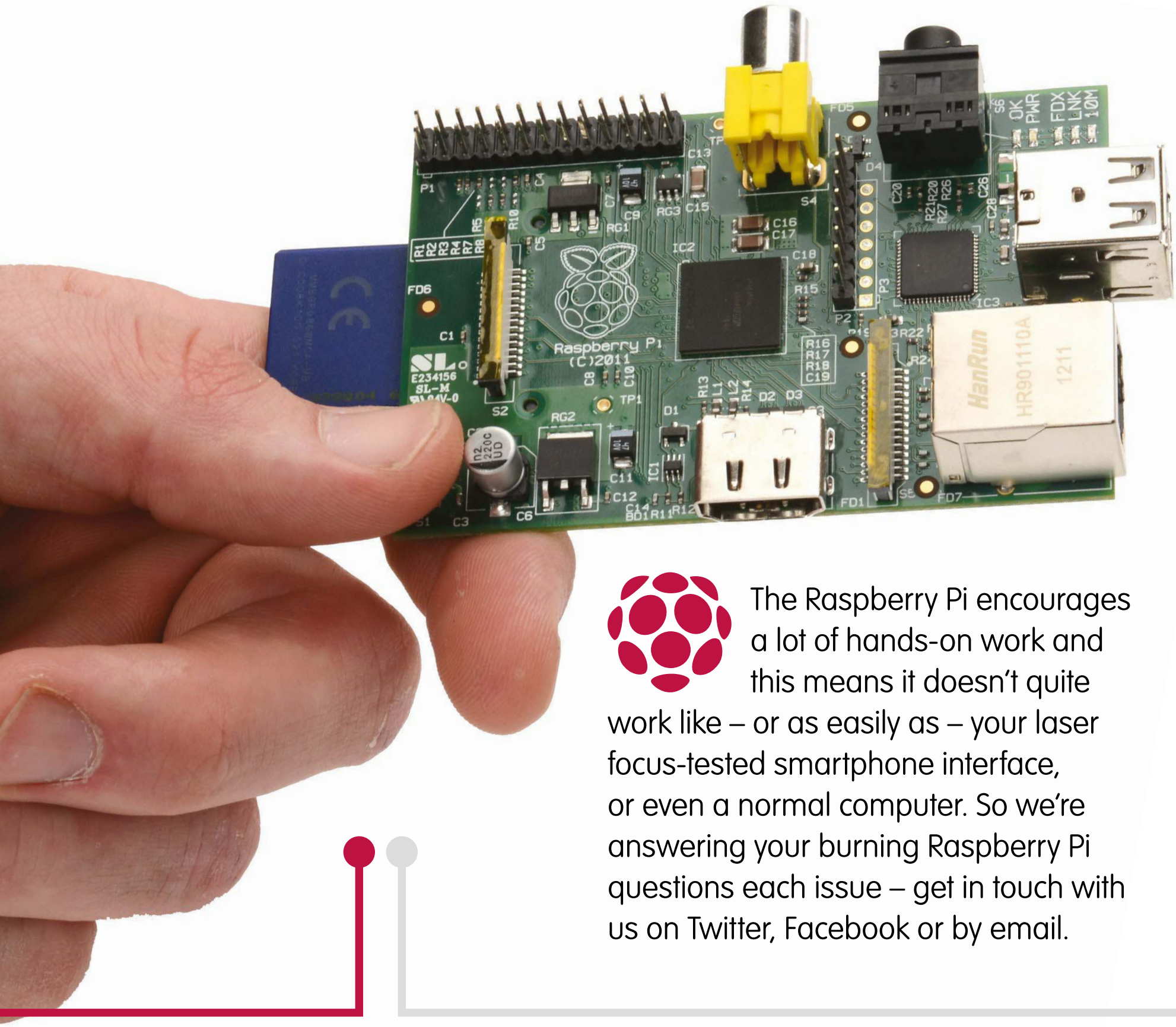
This should allow you to follow whatever security measures are in place when trying to monitor your remote server.

Now that we have figured out how to create a server monitor, there should be no excuse for that mission-critical hardware getting into trouble. You should have all of the up-to-date information you need to be able to manage that system successfully.

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | @ linuxuser@futurenet.com

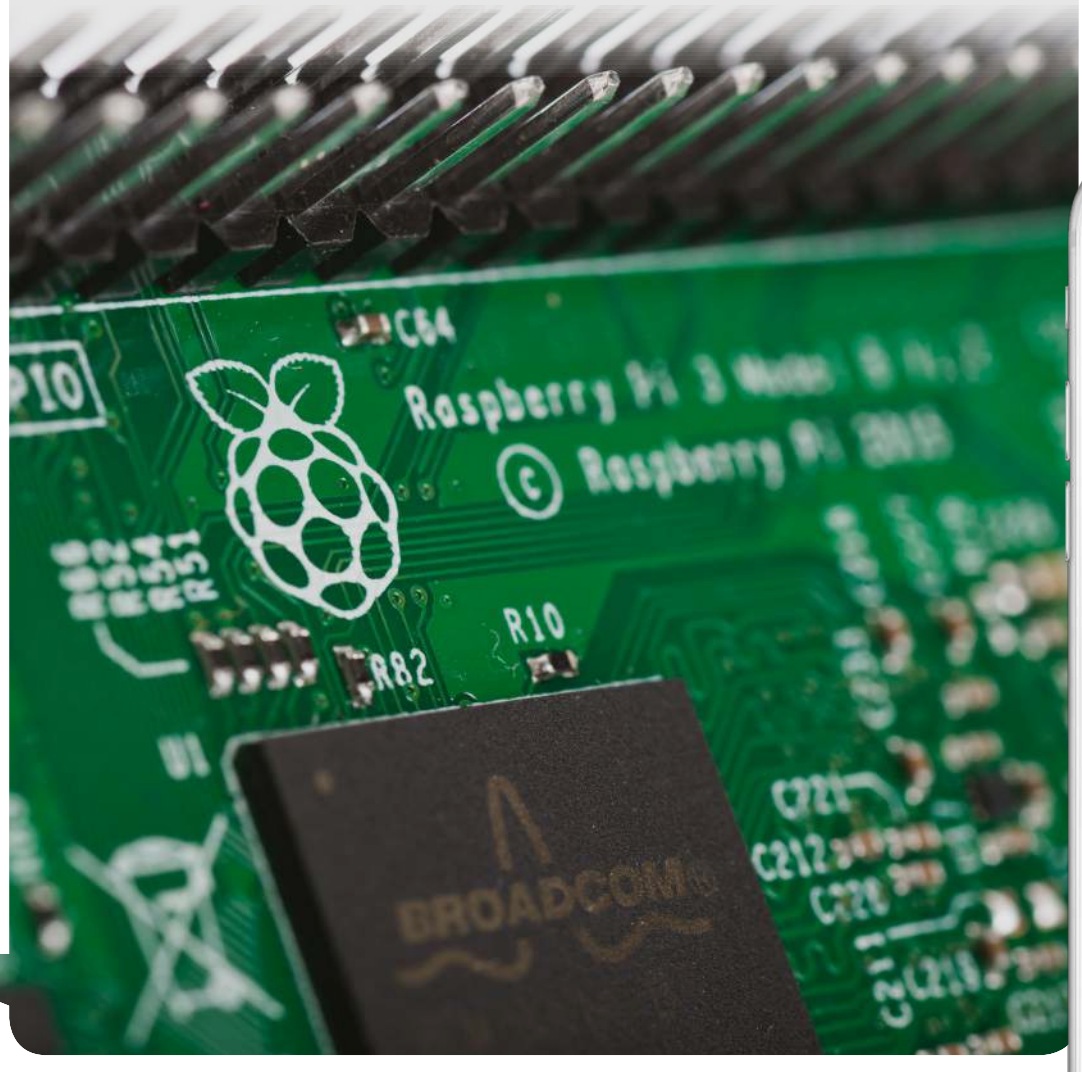


The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easily as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

How can I take screenshots when I'm in the desktop environment

Sh@neTX, Rotherham. via email

You've not told us which distro you're using but that doesn't really matter as there's a neat little trick that you can employ whatever… Simply execute "sudo apt-get install scrot" to install Scrot. Scrot is an ultra-minimal command line screen capturing application and after it's installed you'll be able create and save screenshots of your desktop to the working directory. Scrot will even alllow you to invoke a third-party utility to edit and manipulate the resulting image. It's a highly configurable little app, execute "scrot-h" and take a look at all the options that are open to you.



Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

JUST A SCORE

WHAT'S YOUR JUST A SCORE?

Have you heard of Just A Score? It's a new, completely free app that gives you all the latest review scores. You can score anything in the world, like and share scores, follow scorers for your favourite topics and much more. And it's really good fun!



My Raspberry Pi sometimes shuts down or restarts soon after booting up.
DanGreggs76 via email

If you're getting a flicker from the OK LED before the Raspi fails to start, first check the power supply and SD card as low voltage or a corrupt card can cause this. Make sure the SD Card is making contact with the Raspi as solder residue may have fallen onto the contacts or into the cavity. Clean the contacts and check that they protrude at least 2mm from the lower edge of the holder. If not unclip them by inserting a needle under the contact and pulling up gently. Clean the cavity and reclip.

Can I run Kodi on my Raspberry Pi, and will it be different from the version running on my other devices?
Brian via email

Kodi is a brilliant media player and it's absolutely possible to run it on the Raspberry Pi. At the time of going to press, the current stable version of Kodi for the Pi is Leia, although it's worth noting that you need the version of Kodi designed for the Raspberry Pi – other versions will not run on your Pi. You shouldn't notice any real difference, apart from the fact that it's better not to run Blu-Ray-quality video as it can slow down performance. Some people recommend disabling omxplayer acceleration to improve performance too.



JUST A SCORE
WHAT'S YOUR JUST A SCORE?

You can score absolutely anything on Just A Score. We love to keep an eye on free/libre software to see what you think is worth downloading...

10 Keybase

9 Cinnamon Desktop

8 Tomahawk

4 Anaconda installer

3 FOSS That Hasn't Been Maintained In Years

SCORE ANYTHING
JUST A SCORE

Download on the App Store

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**

Make a Pi-powered virtual reality setup



Get this issue's source code at:

www.linuxuser.co.uk/raspicode

9021

9000